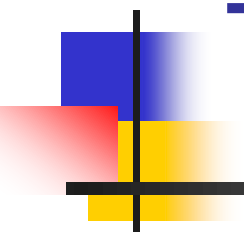
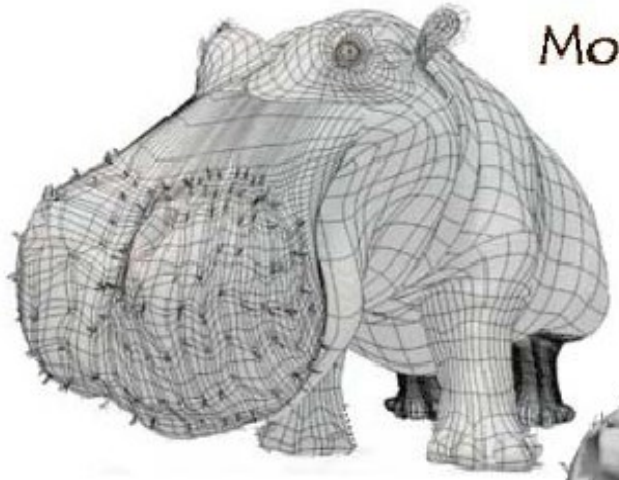


Texture Mapping



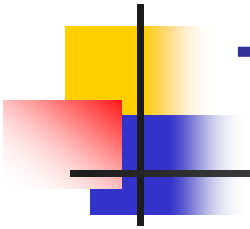
Model

Model + Shading

Model + Shading
+ Textures

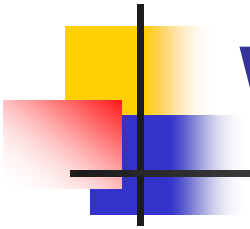
At what point
do things start
looking real?

- For more info on the computer artwork of Jeremy Birn see http://www.3drender.com/jbirn/productions.html



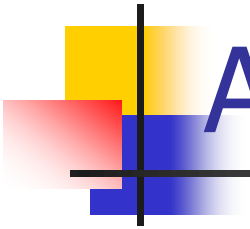
Texture mapping

- To increase surface detail
- Paste a bitmapped image on a surface to provide detail
- To replace original shading with images



Why texture mapping

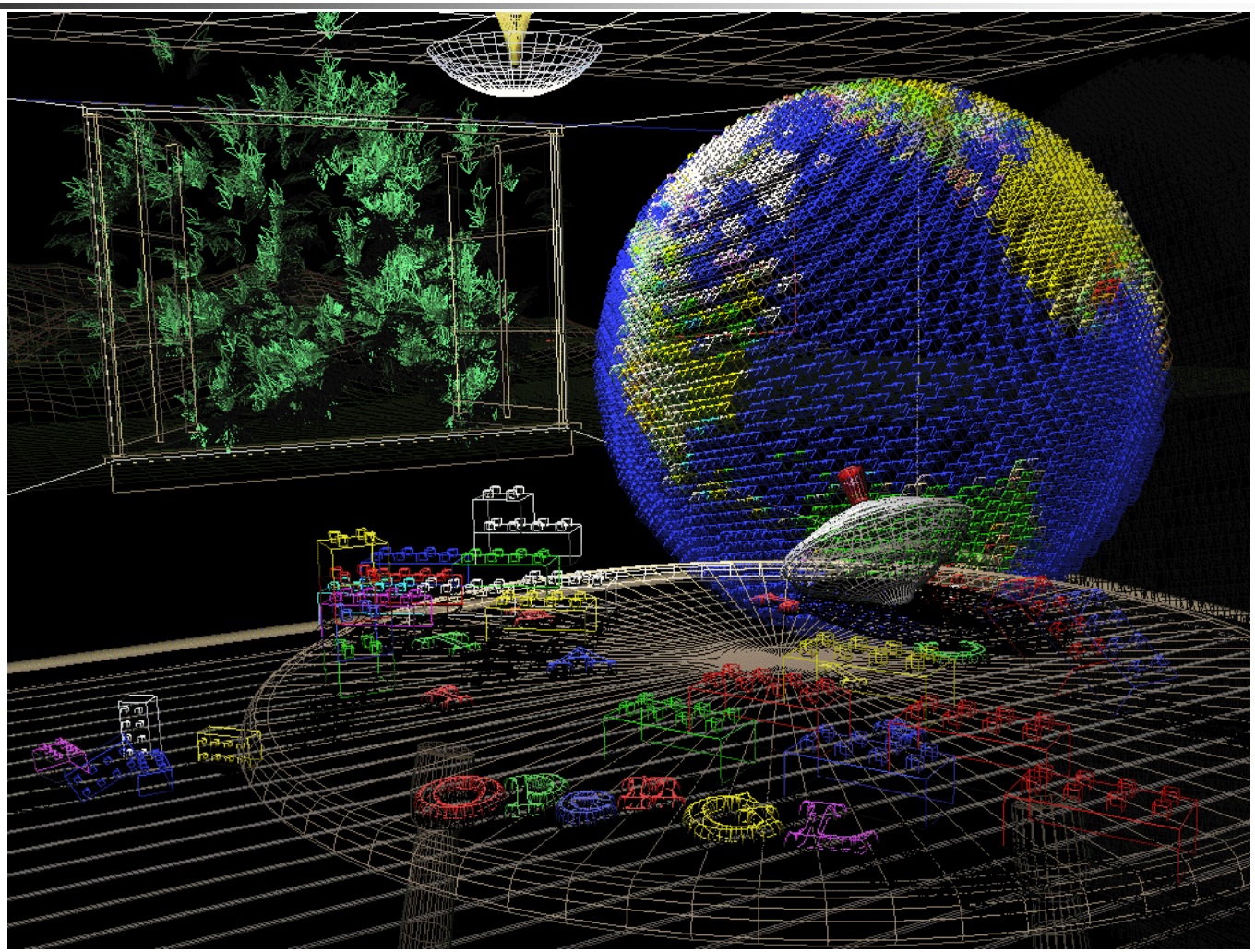
- To get more detail on surface of model
- Expensive solution : Add more polygons to the model
  - Detailed model takes longer to render
  - Requires more space in memory
  - Complex detail cannot be reused on other objects



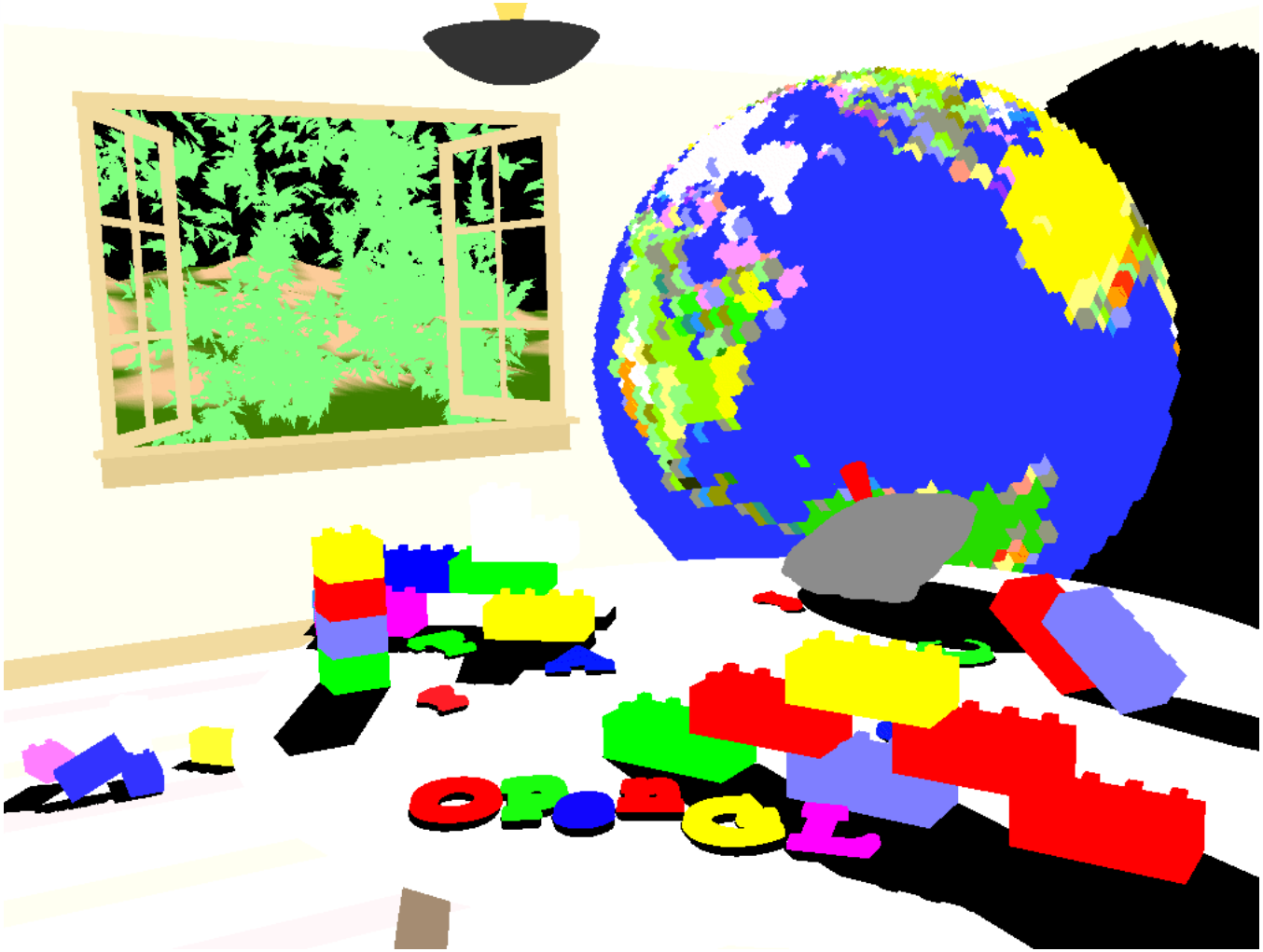
Advantages of texture mapping

- Texture map can be reused for multiple objects
- Texture can be shared (consume less memory) and can be compressed
- Texture maps do not affect the geometry of the objects
- Can be done efficiently

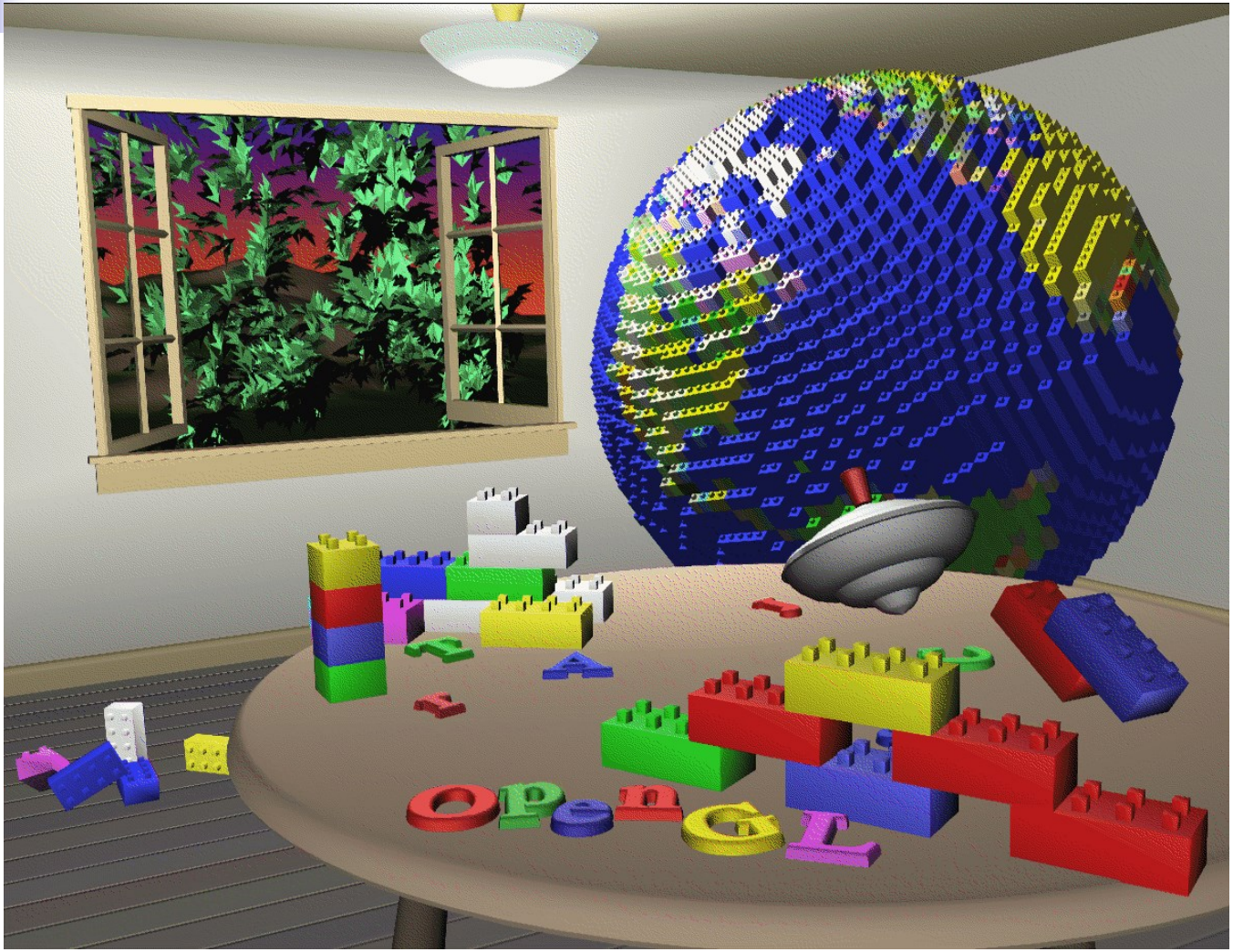
Wire frame



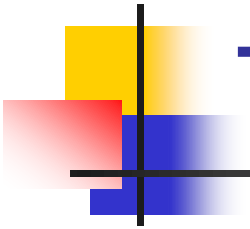
Flat shading



Smooth shading



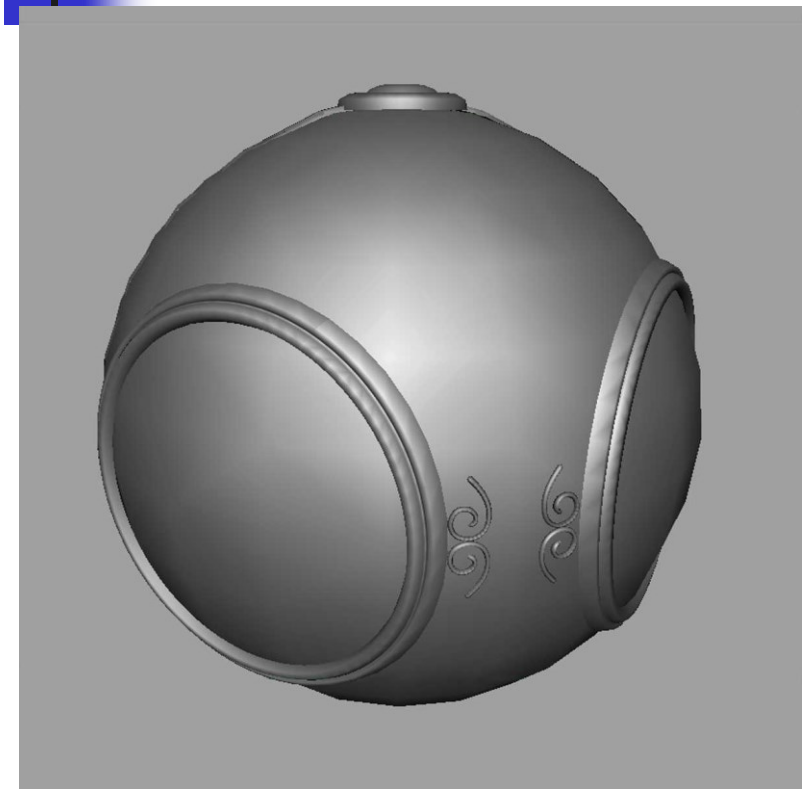
Texture mapped



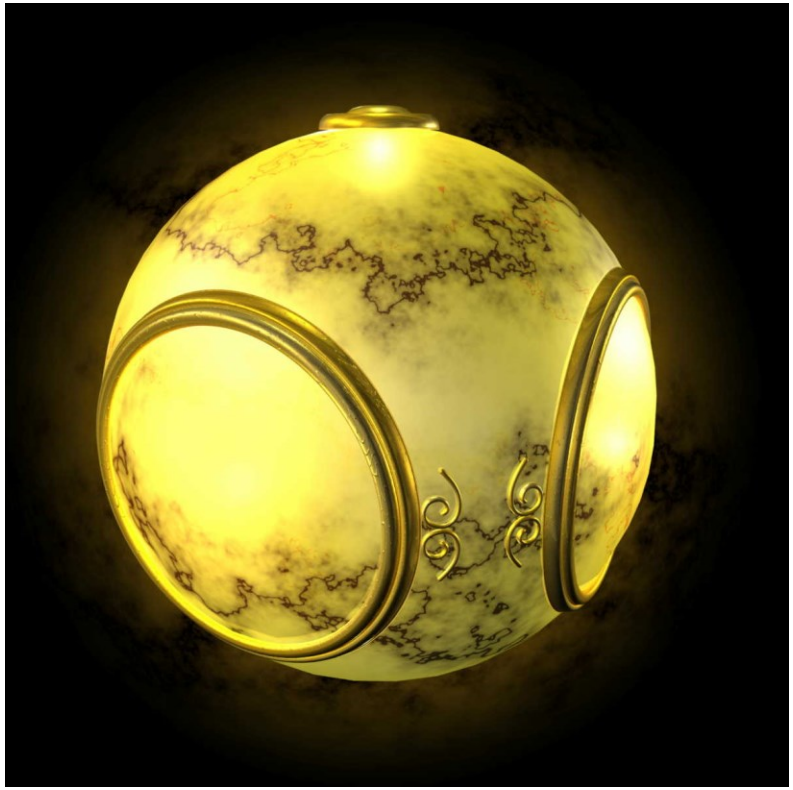
Three Types of Mapping

- Texture Mapping
  - Uses images to fill inside of polygons
- Environment (reflection mapping)
  - Uses a picture of the environment for texture maps
  - Allows simulation of highly specular surfaces
- Bump mapping
  - Emulates altering normal vectors during the rendering process

Texture Mapping



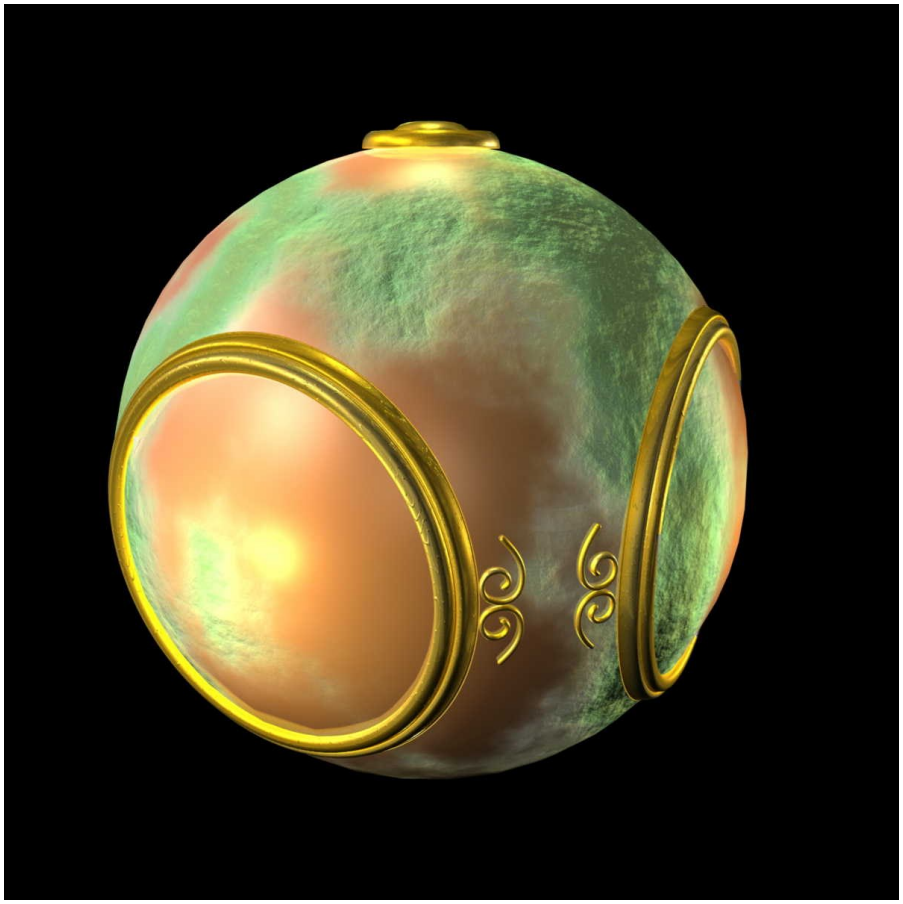
geometric model



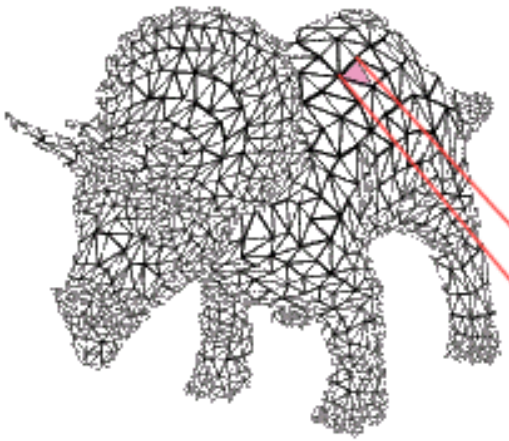
texture mapped

Environment Mapping

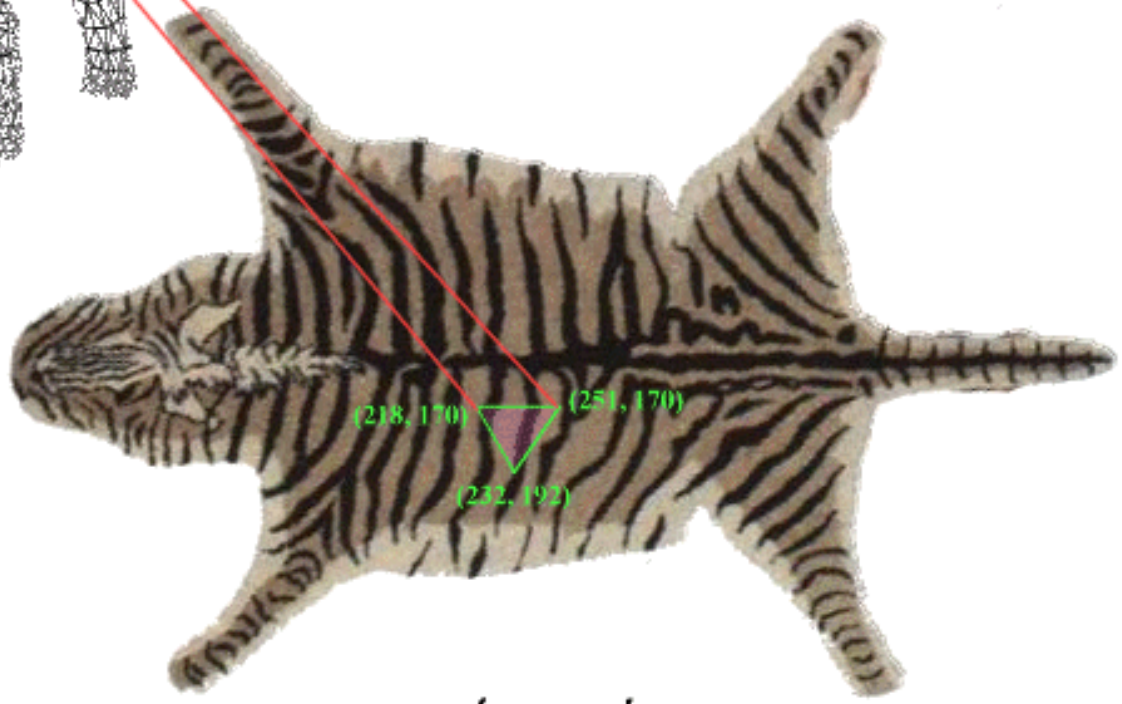
Bump Mapping



Texture Mapping: Rendering



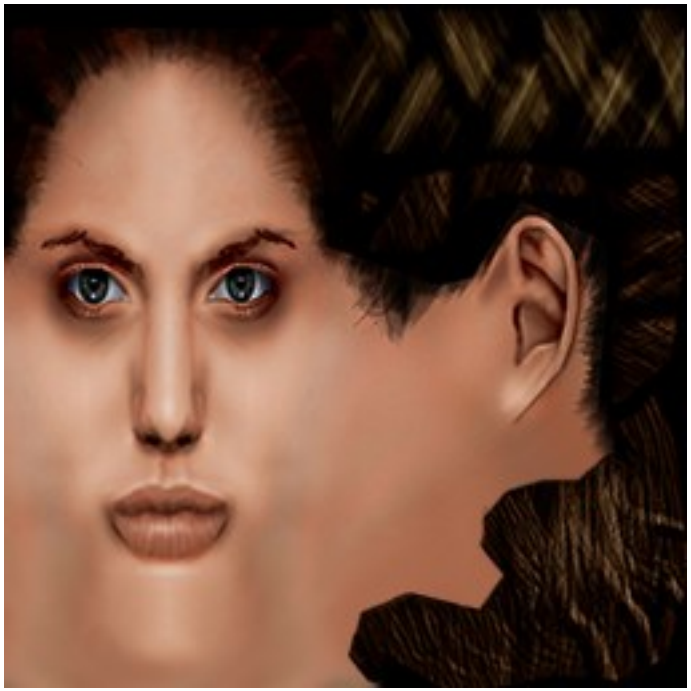
For each triangle in the model establish a corresponding region in the phototexture

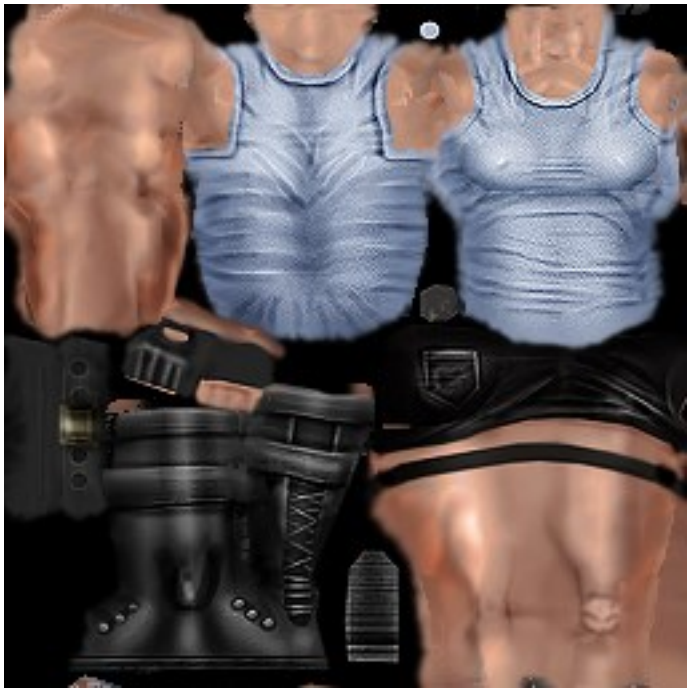


During rasterization interpolate the coordinate indices into the texture map

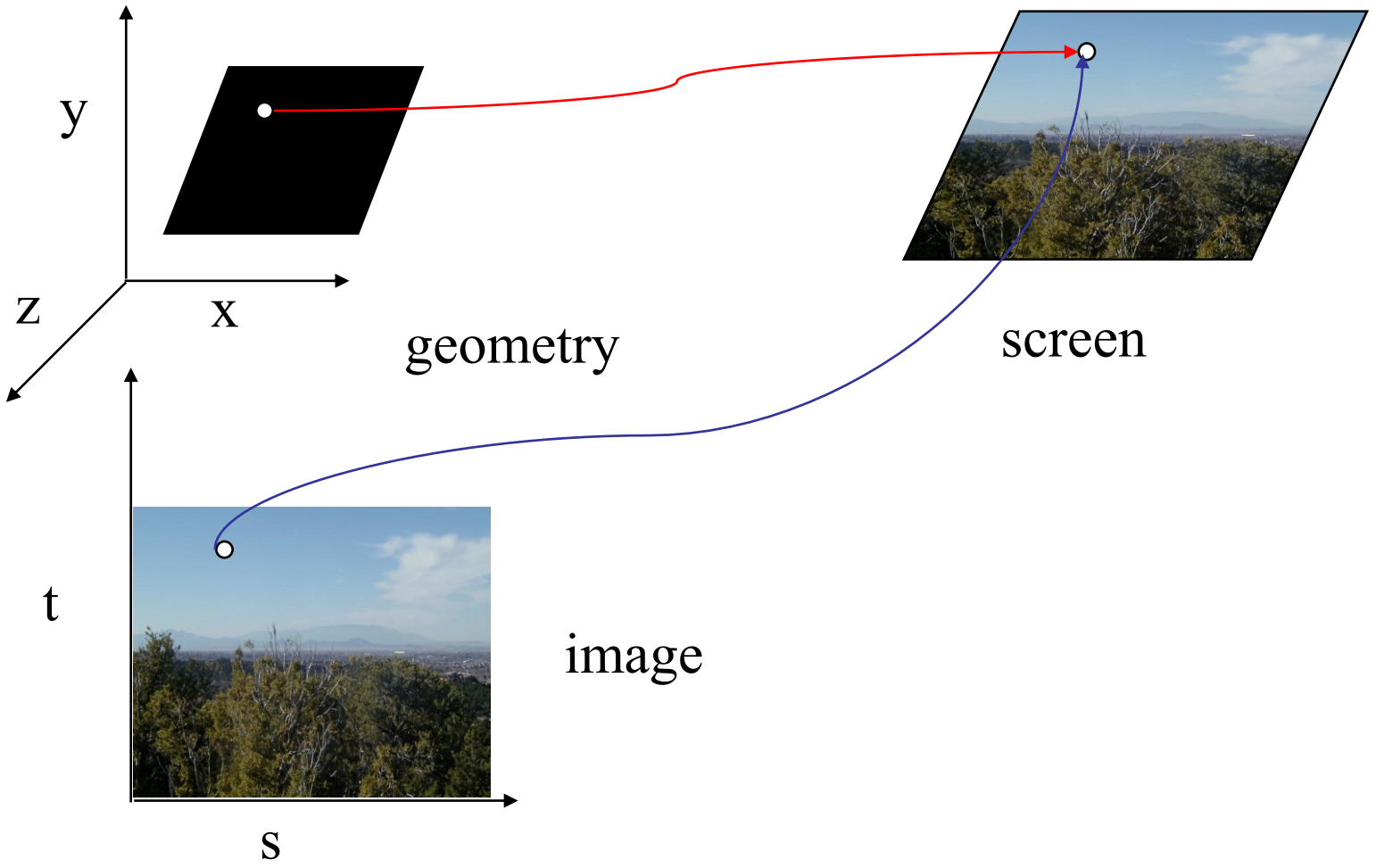
User-Generated Mappings

- For complex 3-D objects, mapping textures is still something of an art…so we often let the user do it





Texture Mapping



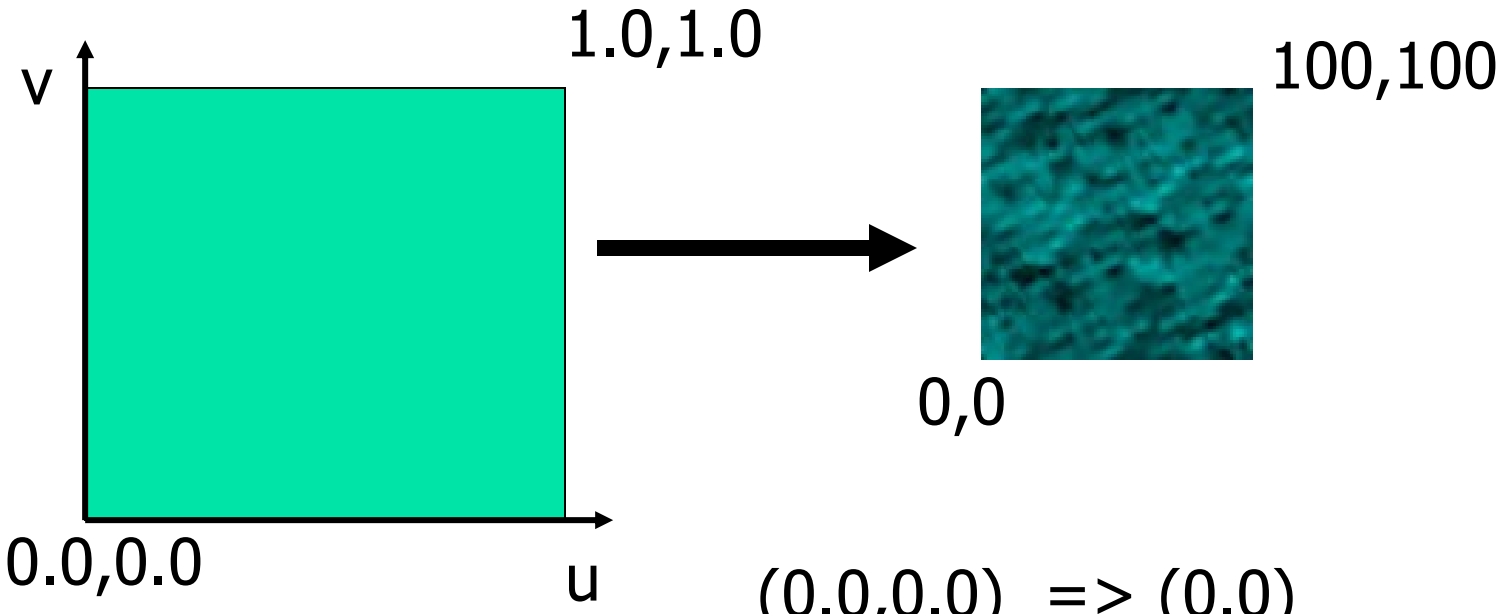
Mapping function

- Points on a surface in object-space

Mapping function

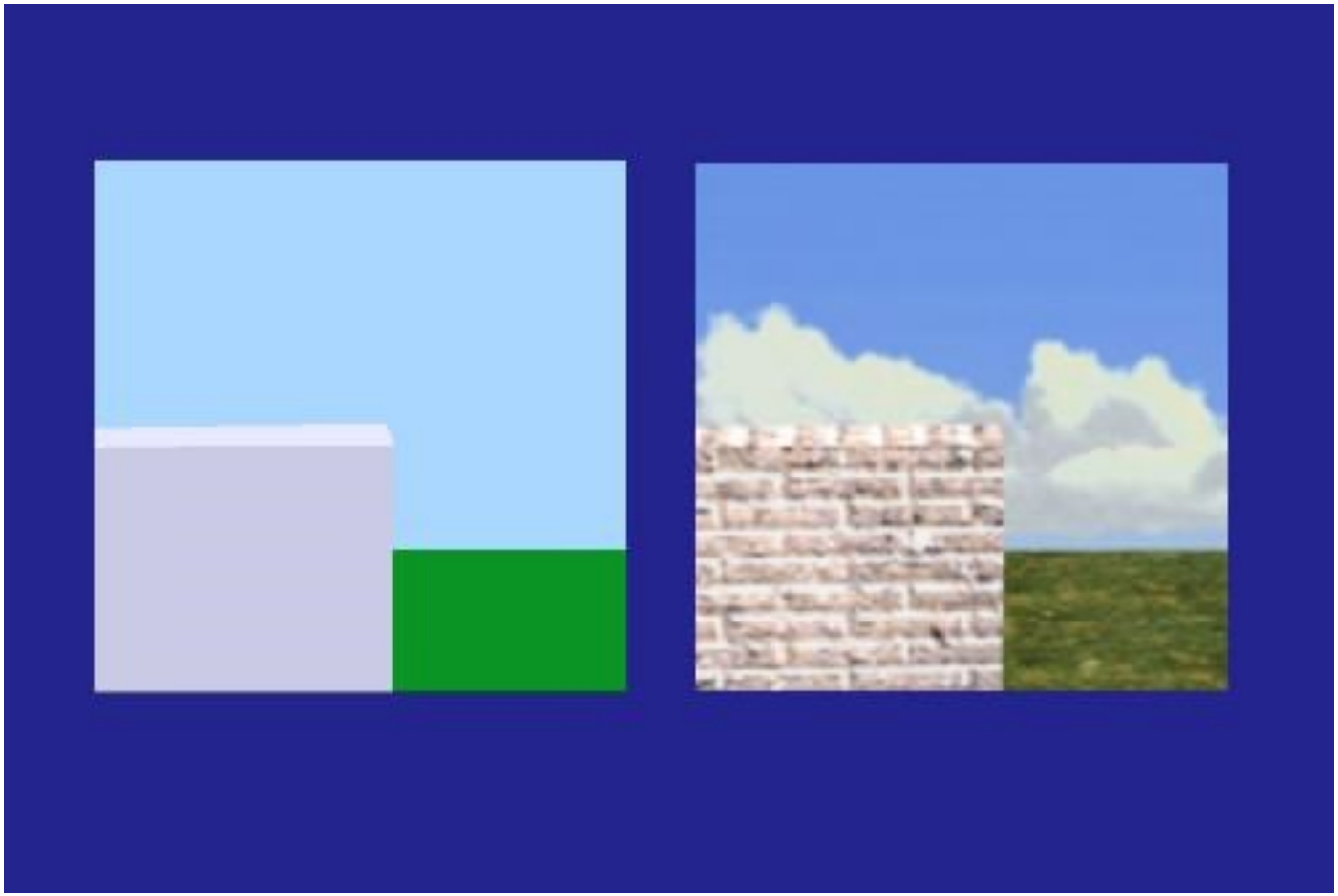
- Pixel values from a texture map images

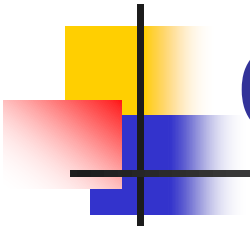
Basic Idea



(0.0,0.0) => (0.0)
(1.0,1.0) => (100,100)
(0.5,0.5) => (50,50)

It’s simple if all surfaces are flat and rectangle



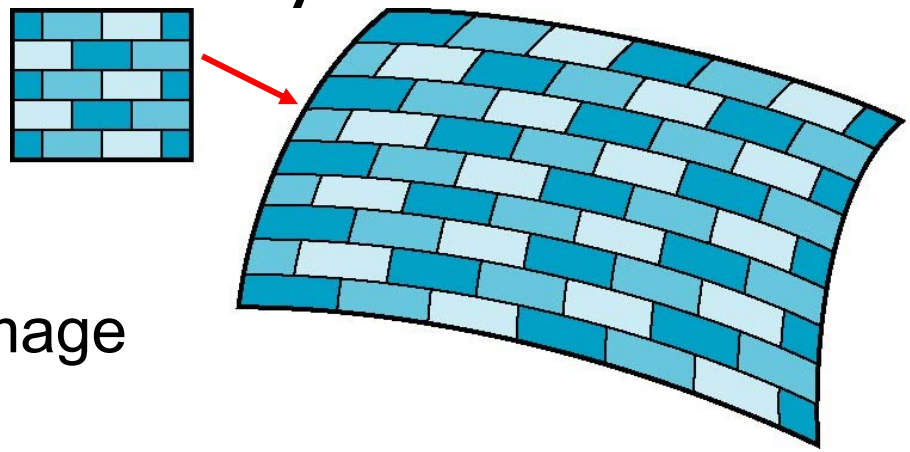


Concept is simple but …

- 2D (texture map) -> 3D (object surface) is not easy because ...
- Consideration 1:
  - texture map is flat,
  - object surface is arbitrary.
  - Distortion is unavoidable.

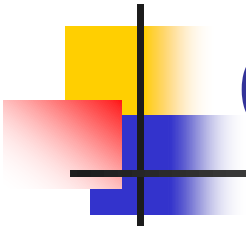
Is it simple?

- Although the idea is simple---map an image to a surface---there are 3 or 4 coordinate systems involved



2D image

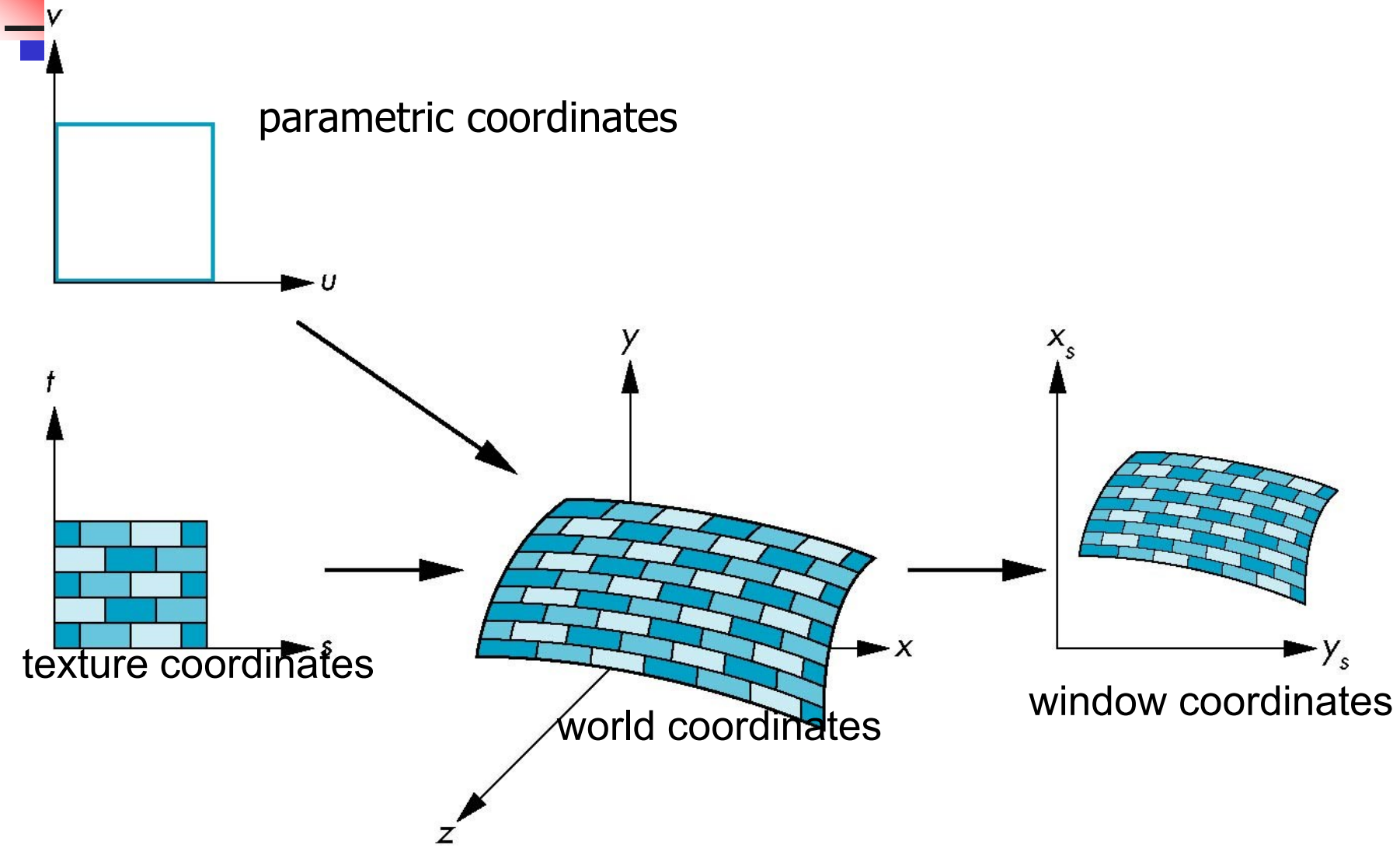
3D surface



Coordinate Systems

- Parametric coordinates
  - May be used to model curves and surfaces
- Texture coordinates
  - Used to identify points in the image to be mapped
- Object or World Coordinates
  - Conceptually, where the mapping takes place
- Window Coordinates
  - Where the final image is really produced

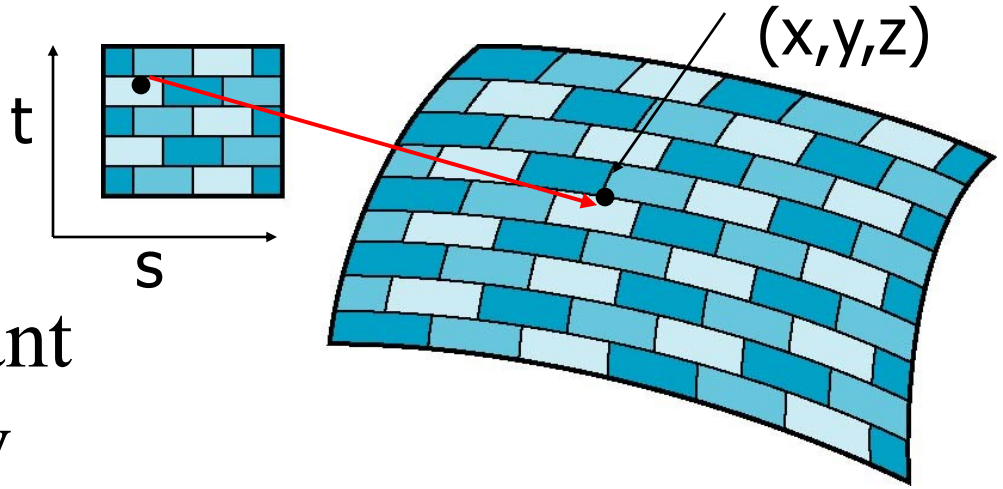
Texture Mapping



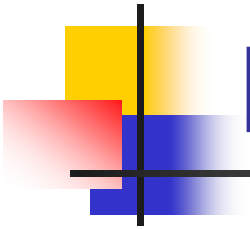
Mapping Functions

- Basic problem is how to find the maps
- Consider mapping from texture coordinates to a point a surface
- Appear to need three functions

$$x = x(s,t)$$
$$y = y(s,t)$$
$$z = z(s,t)$$



- But we really want to go the other way



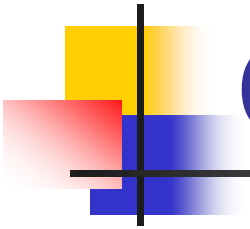
Backward Mapping

- We really want to go backwards
  - Given a pixel, we want to know to which point on an object it corresponds
  - Given a point on an object, we want to know to which point in the texture it corresponds
- Need a map of the form

$$s = s(x,y,z)$$

$$t = t(x,y,z)$$

- Such functions are difficult to find in general

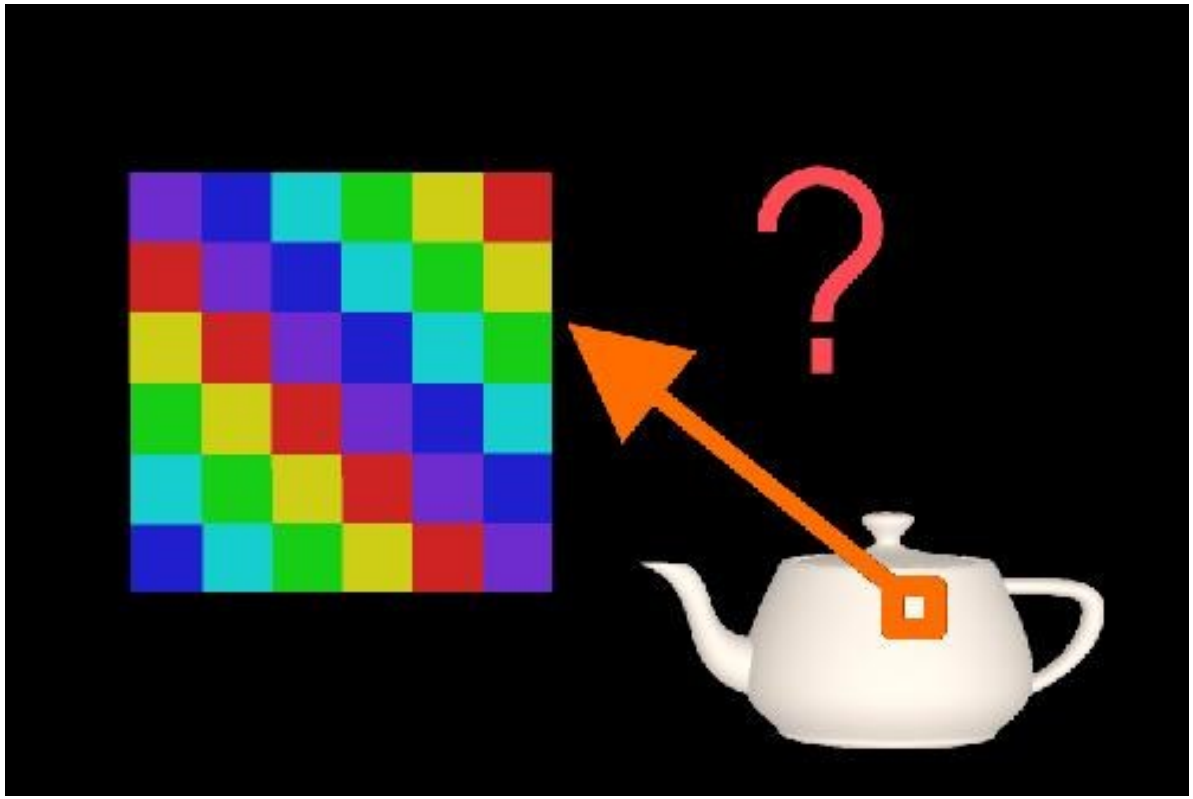


Object space, texture space

- The unit uv square applies to object space
- Texture space referenced with (s,t) coordinates
- Finding a mapping from (u,v) to (s,t)

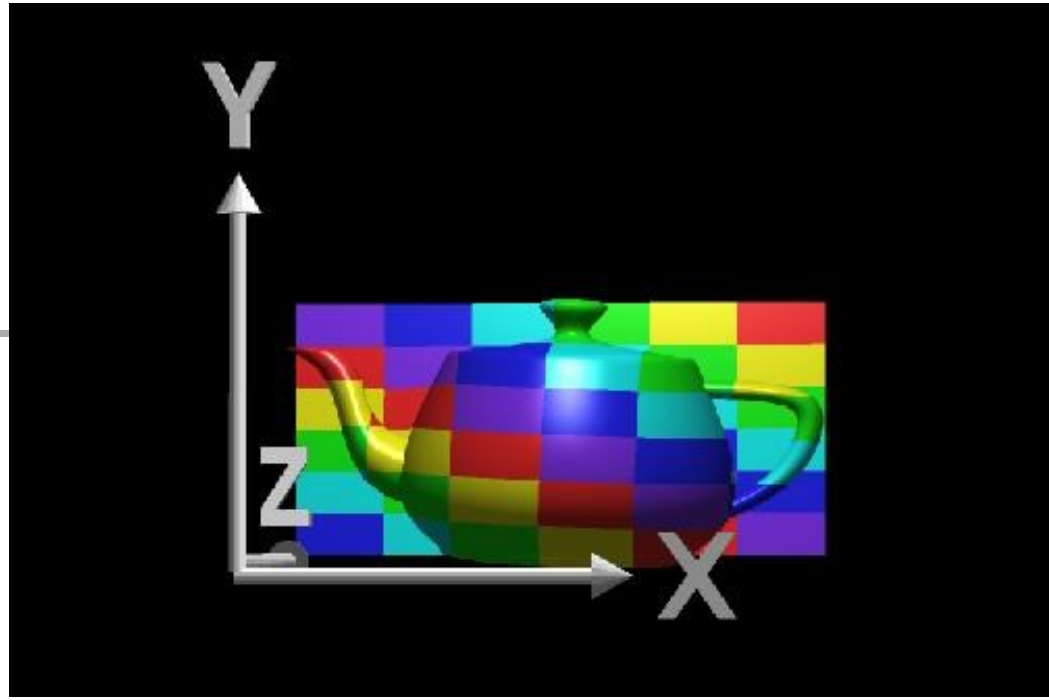
Proper mapping

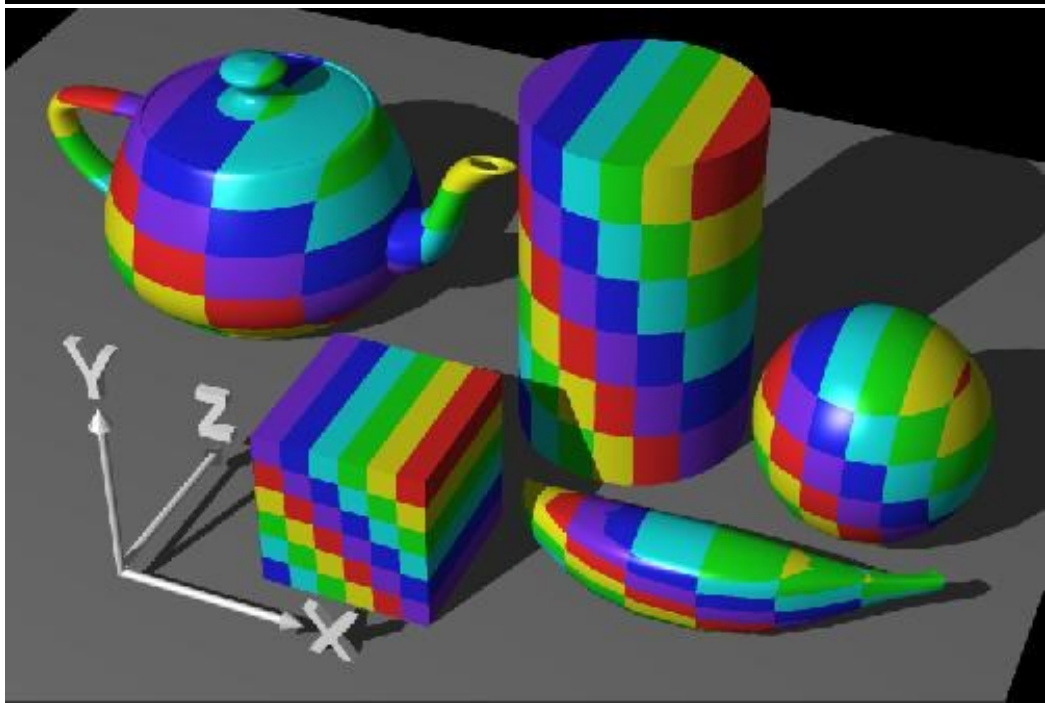
- In two-dimensional texture mapping, we have to decide how to paste the image on to an object. In other words, for each pixel in an object, we encounter the question, "Where do I have to look in the texture map to find the color?" To answer this question, we consider two things: map *shape* and map *entity.*



1st attempt

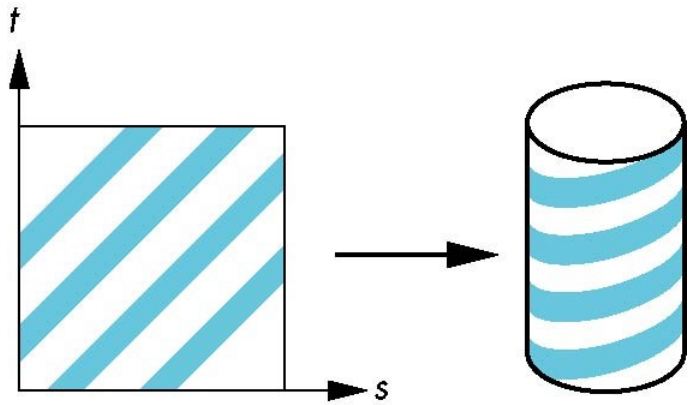
- In this case, the component that was thrown away was the *z*-coordinate.

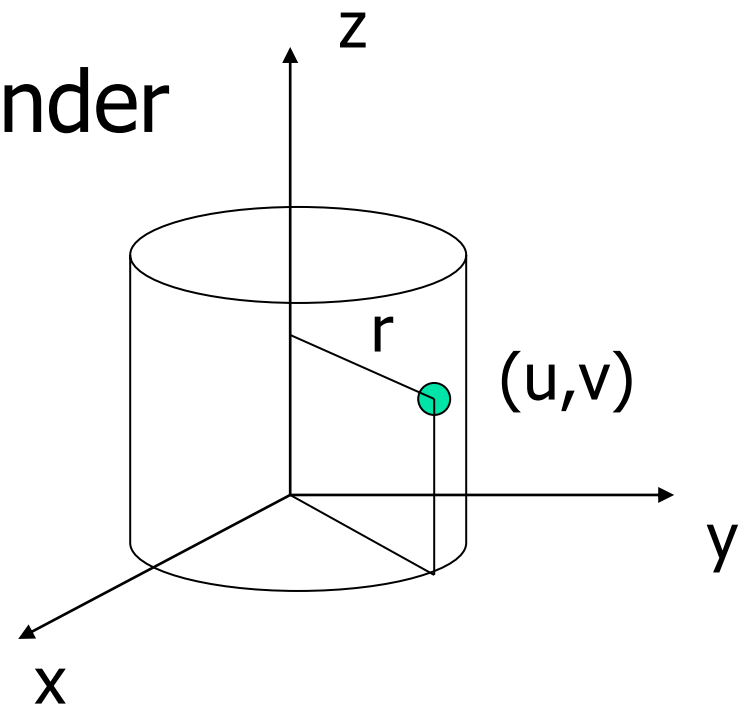


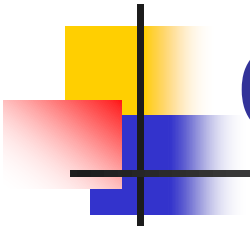


Two-part mapping

- One solution to the mapping problem is to first map the texture to a simple intermediate surface
- Example: map to cylinder







Cylindrical Mapping

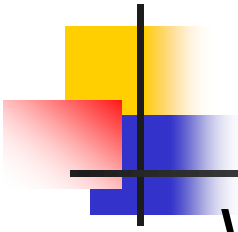
parametric cylinder

$$x = r \cos 2\pi u$$
$$y = r \sin 2\pi u$$
$$z = v/h$$

maps rectangle in u,v space to cylinder
of radius r and height h in world coordinates

$$s = u$$
$$t = v$$

maps from texture space



Spherical Map

We can use a parametric sphere

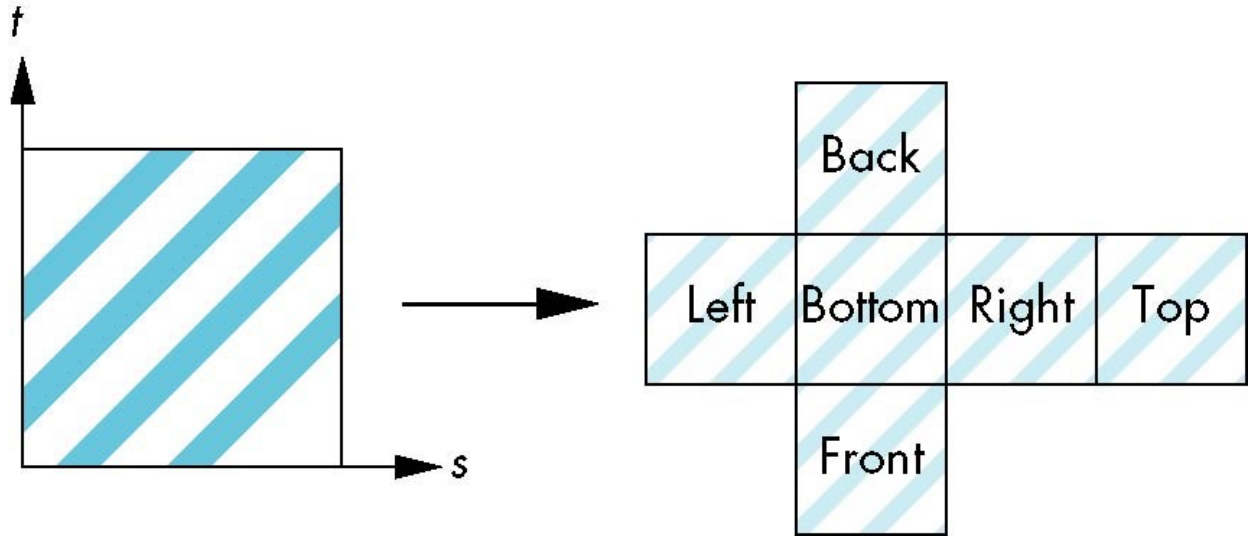
$$x = r \cos 2\pi u$$
$$y = r \sin 2\pi u \cos 2\pi v$$
$$z = r \sin 2\pi u \sin 2\pi v$$

in a similar manner to the cylinder
but have to decide where to put
the distortion

Spheres are used in environmental maps

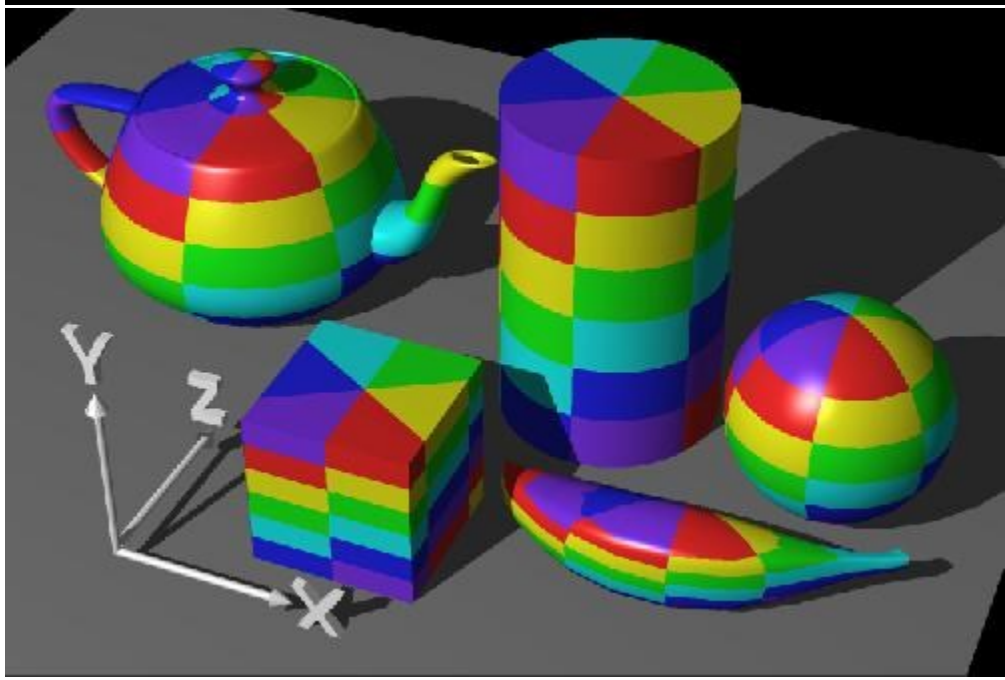
Box Mapping

- Easy to use with simple orthographic projection
- Also used in environment maps



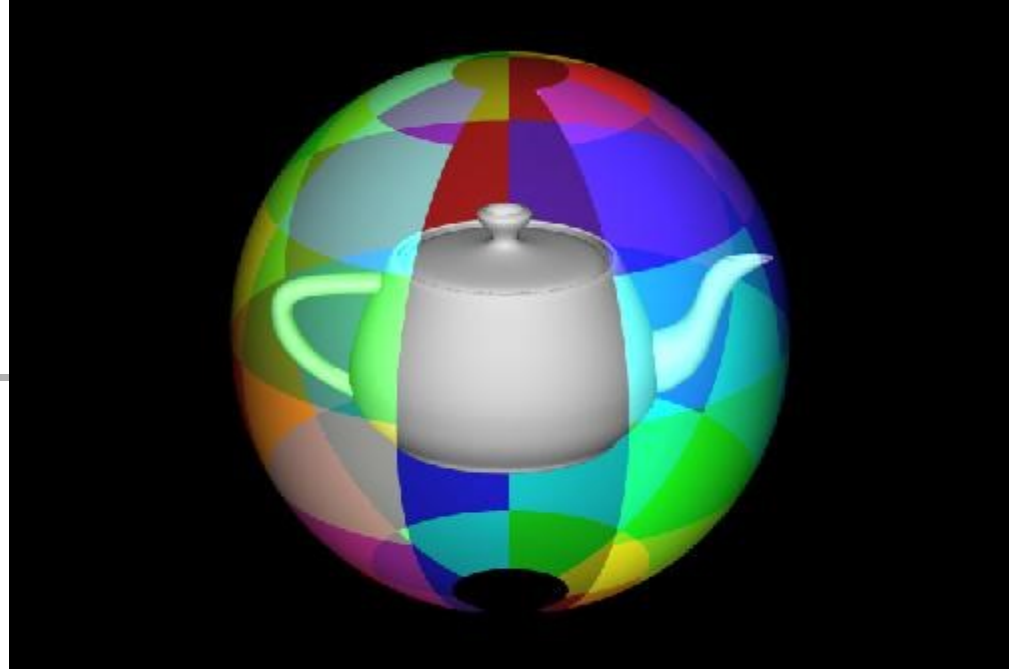
Cylinder

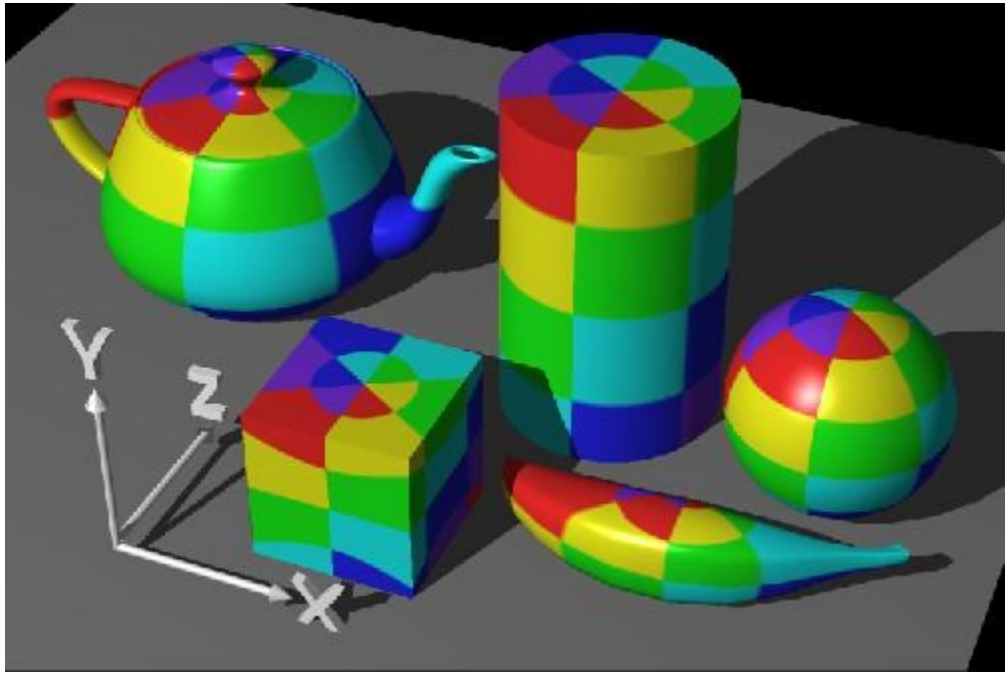
- The texture-mapped objects in this image have a cylindrical map shape, and the cylinder's axis is parallel to the *z*-axis. At the smallest *z*-position on each object, note that the squares of the texture pattern become squeezed into "pie slices". This phenomenon occurs at the greatest *z* position as well. When the cylinder's axis is parallel to the *z*-axis, you'll see "pie slices" radiating out along the *x*- and *y*- axes.



Sphere

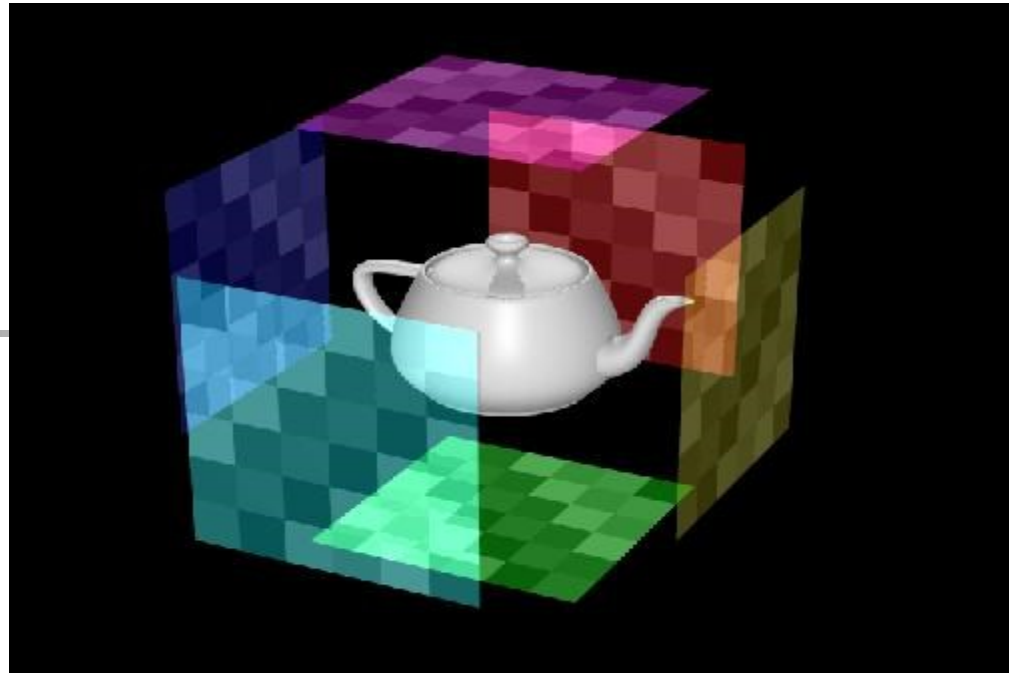
- The objects have a map shape of a sphere, and the poles of the sphere are parallel to the *y*-axis. At the object's "North Pole" and "South Pole", the squares of the texture map become squeezed into pie-wedge shapes. Compare this slide to slide 12 which has a map shape of a cylinder. Both map shapes have the pie-wedge shapes at the poles, but there is a subtle difference at the object's "equator". The spherical mapping stretches the squares in the texture map near the equator, and squeezes the squares as the longitude reaches a pole.

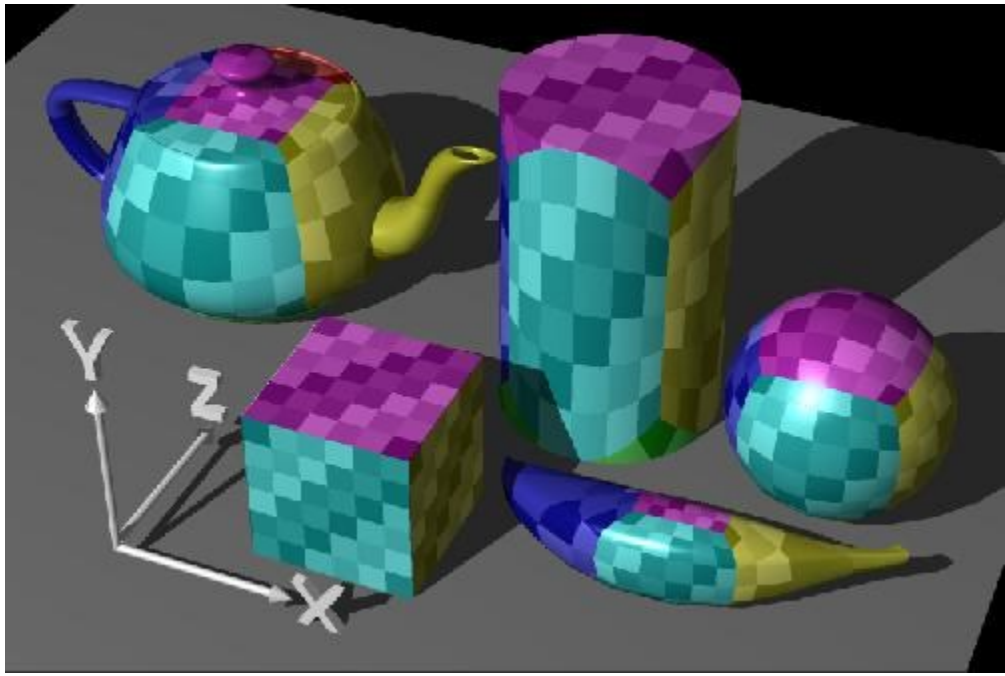




box

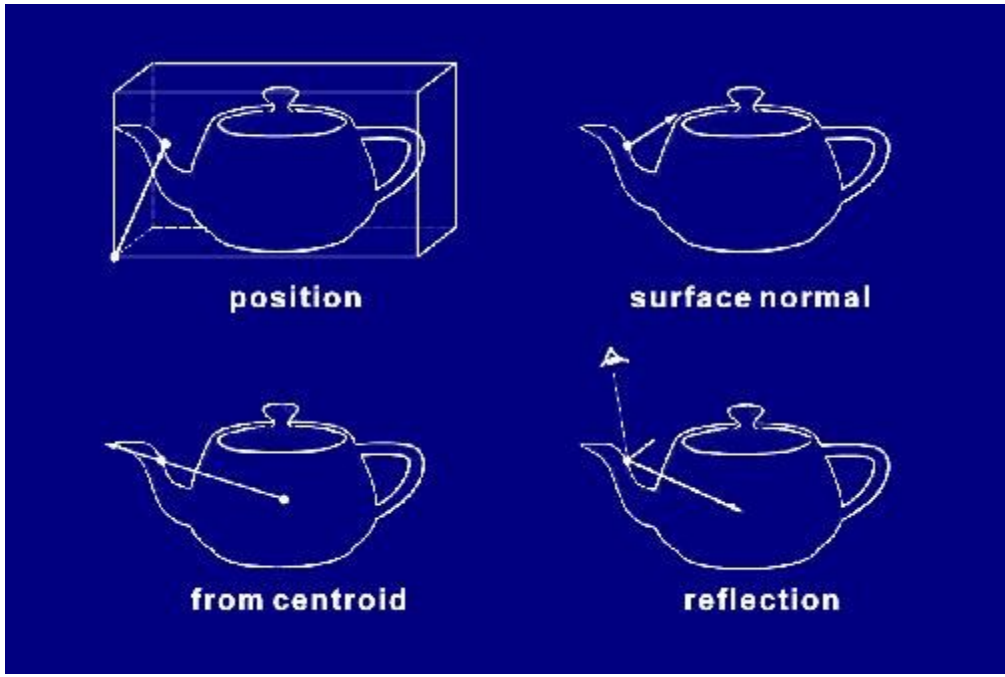
- Using a box as the map shape is similar to planar mapping. Instead of using one texture map, box mapping uses six -- one each for the left, right, front, back, top and bottom sides of the object. To texture map the front and back sides, we eliminate the *z*-component of an object's point and use the remaining *x*- and *y*-components to locate the color in the corresponding texture maps.





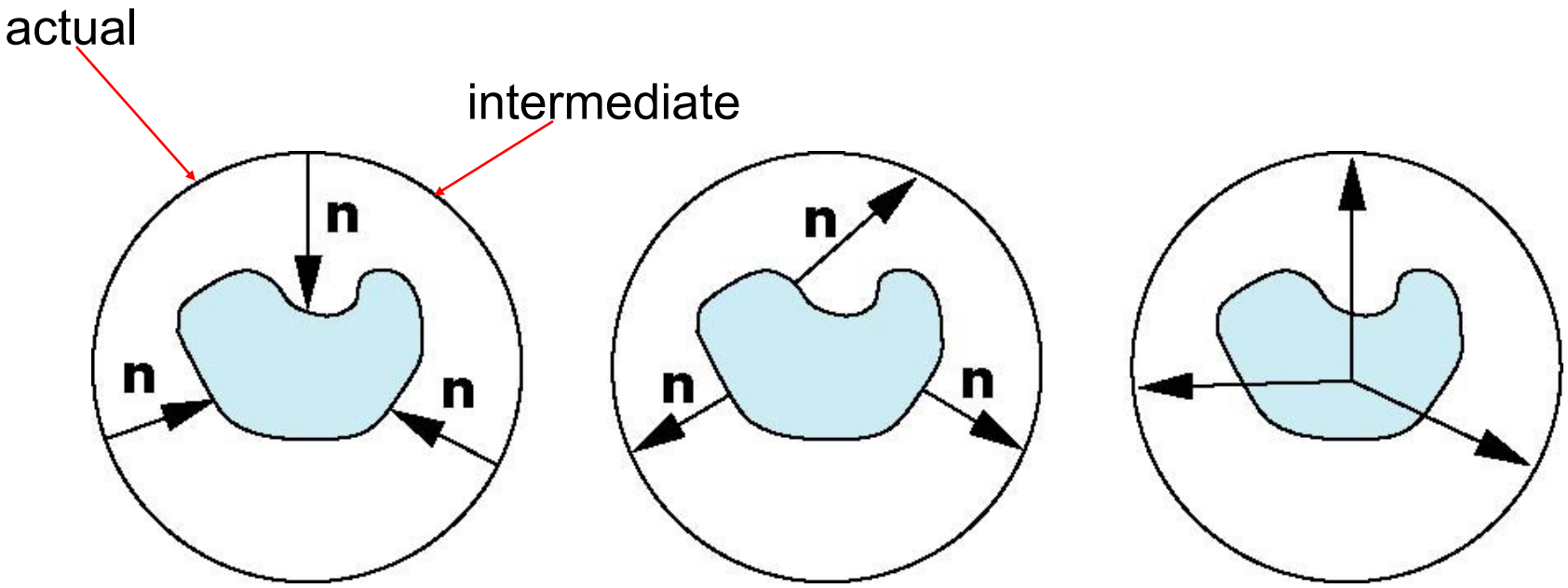
How to map

- When discussing map shape, we talked about taking an (*x,y,z*) value from the object and converting in various ways, but we didn't mention what that value was. The map entity determines what we use as the (*x,y,z*) value. Commonly-used map entities are 1) a point on the object relative to the object's bounding box, 2) the surface normal at the point being rendered, 3) a vector running from the object's centroid through the point, and 4) the reflection vector at the current point. Remember that the reflection vector depends not only on the position of the point and its normal, but on the position of the viewer.

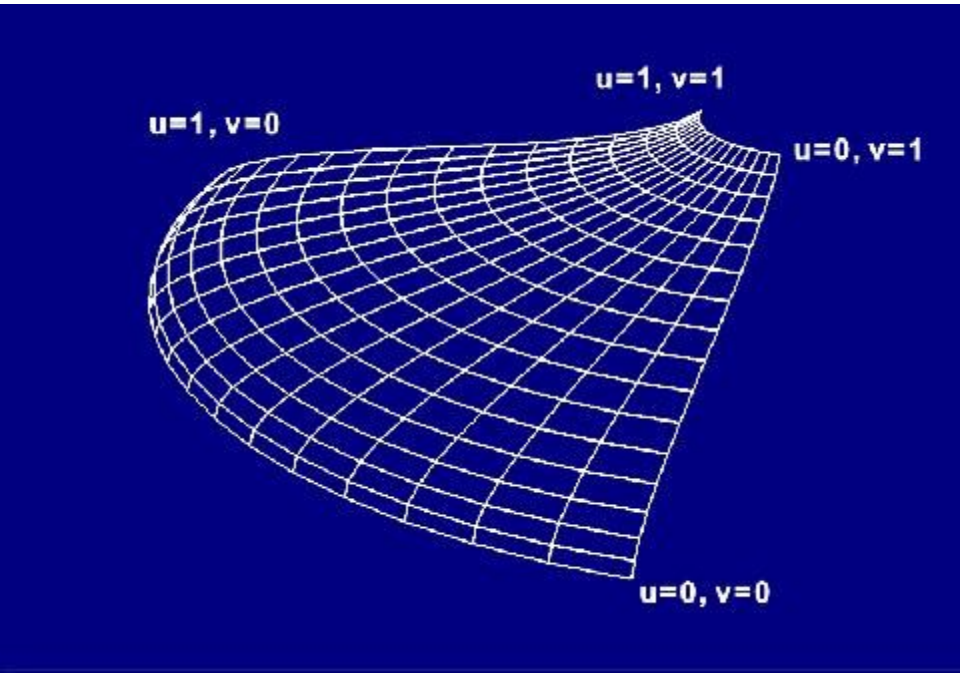


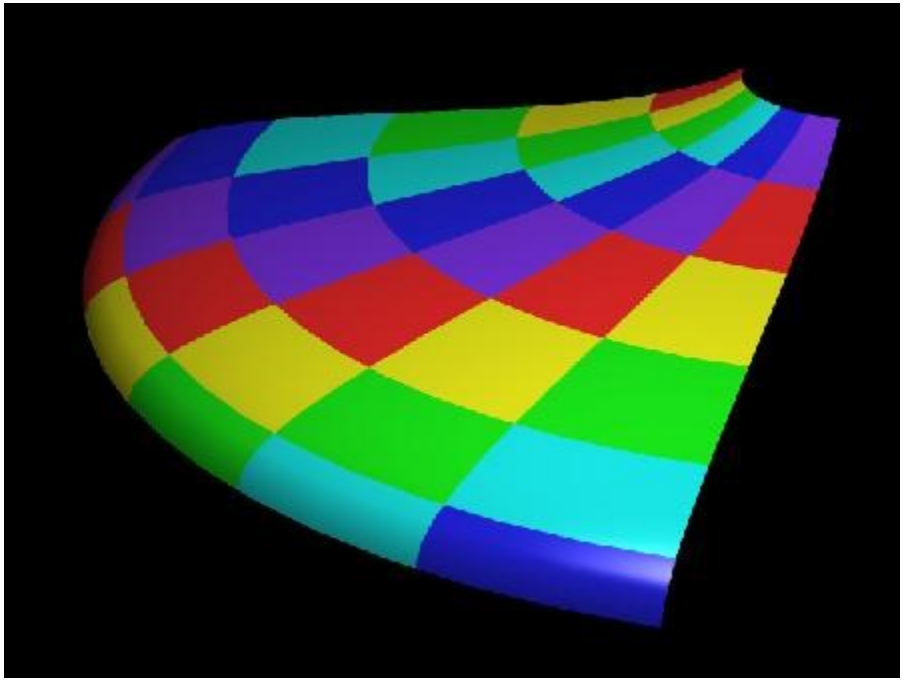
Second Mapping

- Map from intermediate object to actual object
  - Normals from intermediate to actual
  - Normals from actual to intermediate
  - Vectors from center of intermediate



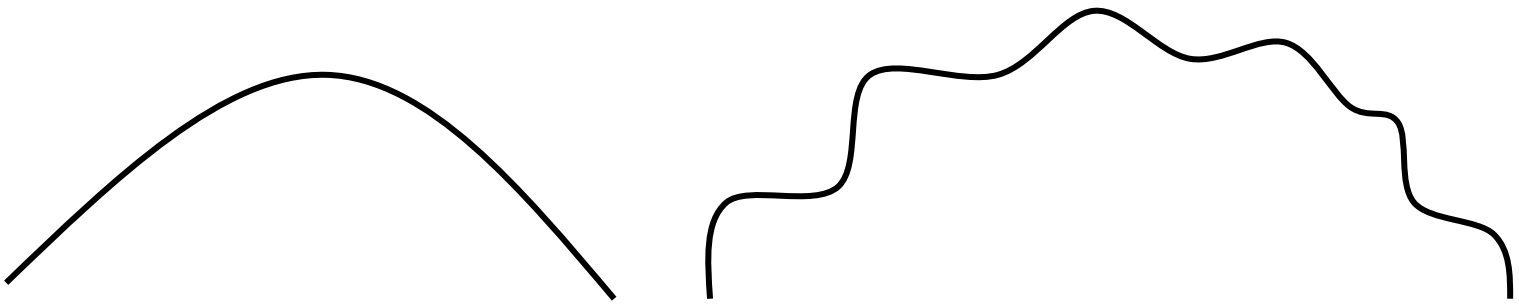
Mapping on Bezier



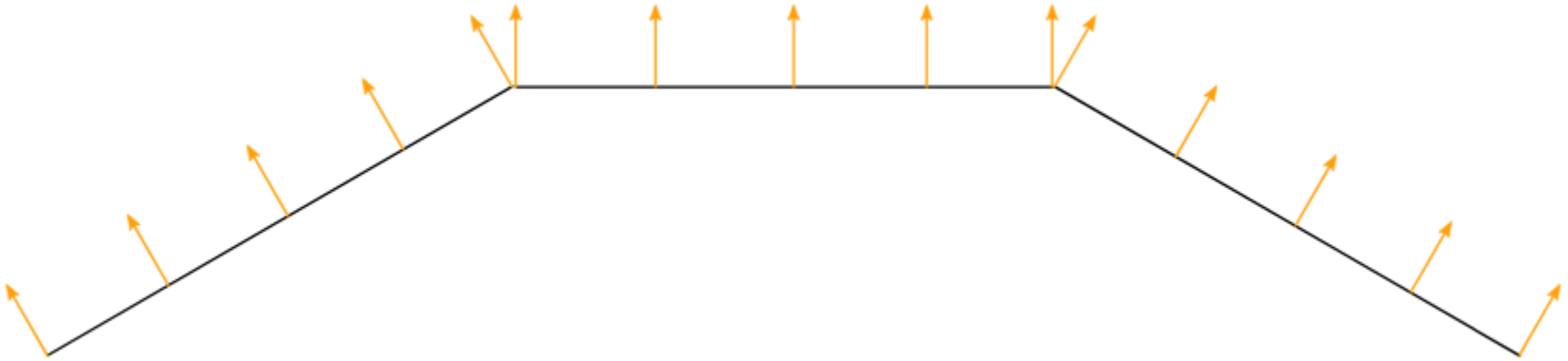


Bump Mapping

- Evaluating illumination with perturbed surface normals
- Geometry doesn't change, only applying another layer of fake surface normals

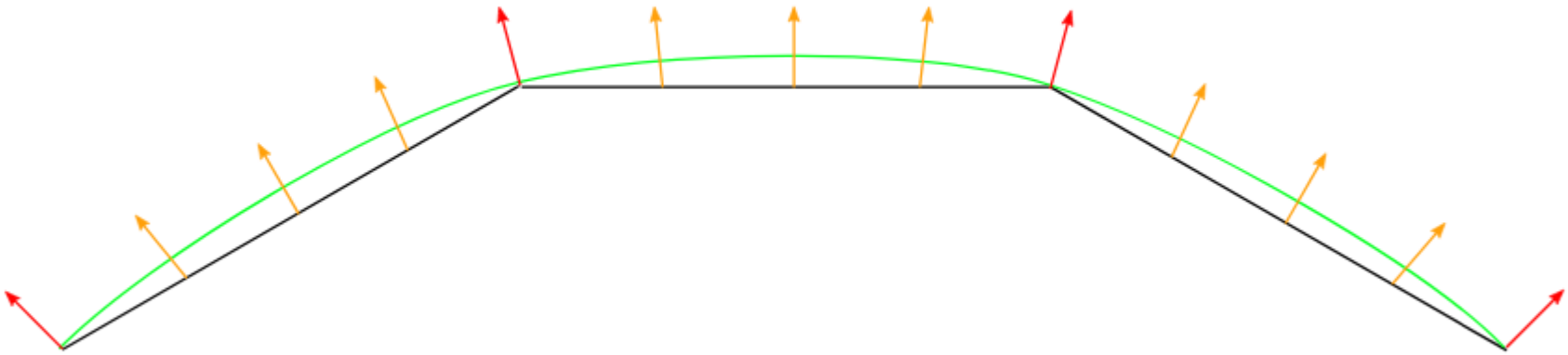


Comparison – flat shading



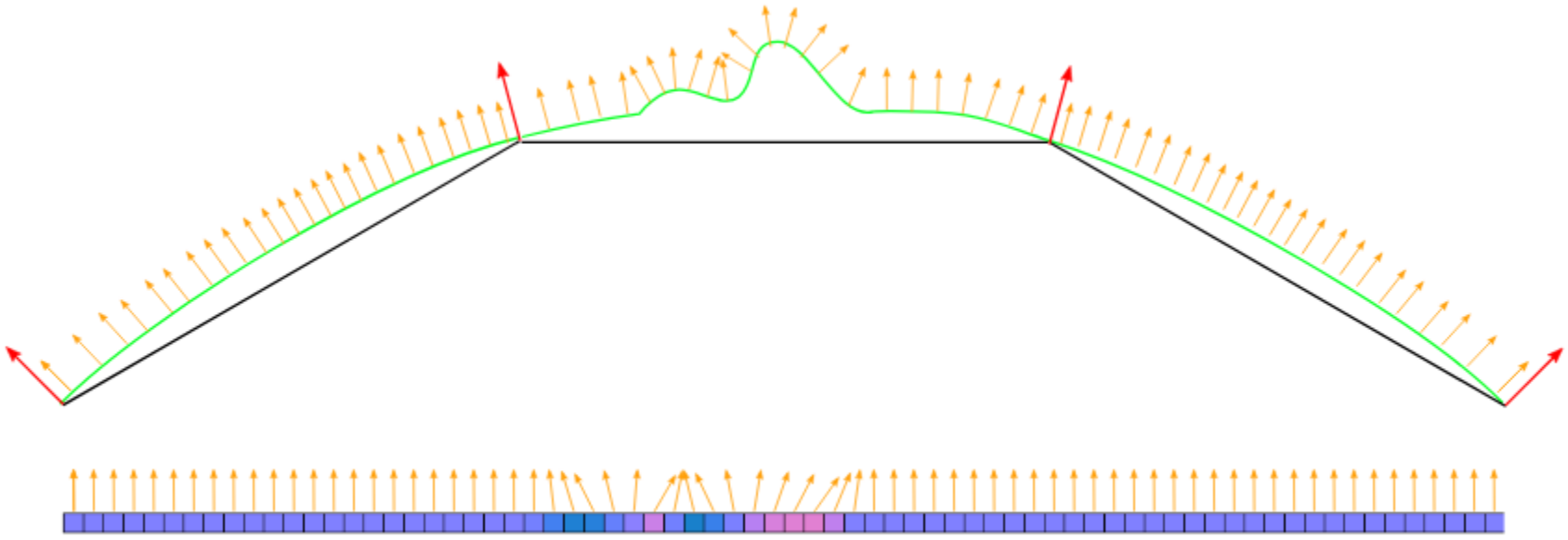
Flat shading on three polygons, viewed as a 2D diagram

Comparison – smooth shading



Smooth shading on three polygons, viewed as a 2D diagram

Comparison – bump mapping

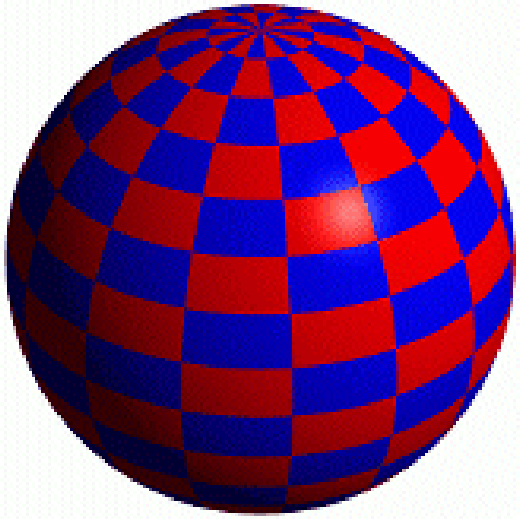


Normal mapping across three polygons, viewed as a 2D diagram

Bump Mapping

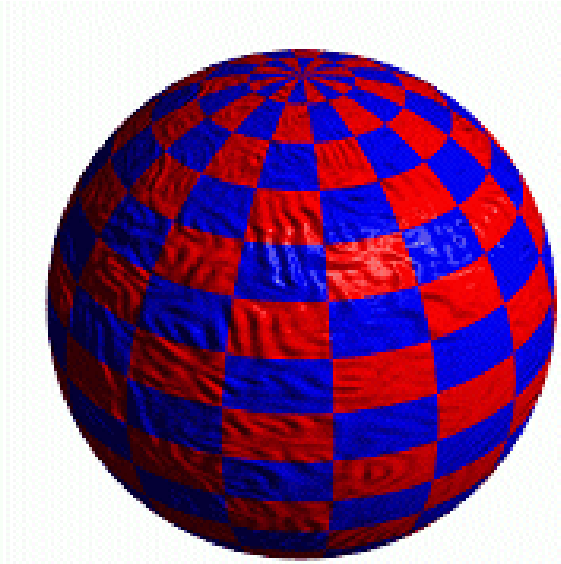
Textures can be used to alter the surface normal of an object. This does not change the actual shape of the surface -- we are only shading it as if it were a different shape! This technique is called *bump mapping*. The texture map is treated as a single-valued height function. The value of the function is not actually used, just its partial derivatives. The partial derivatives tell how to alter the true surface normal at each point on the surface to make the object appear as if it were deformed by the height function.

Since the actual shape of the object does not change, the silhouette edge of the object will not change. Bump Mapping also assumes that the Illumination model is applied at every pixel (as in Phong Shading or ray tracing).



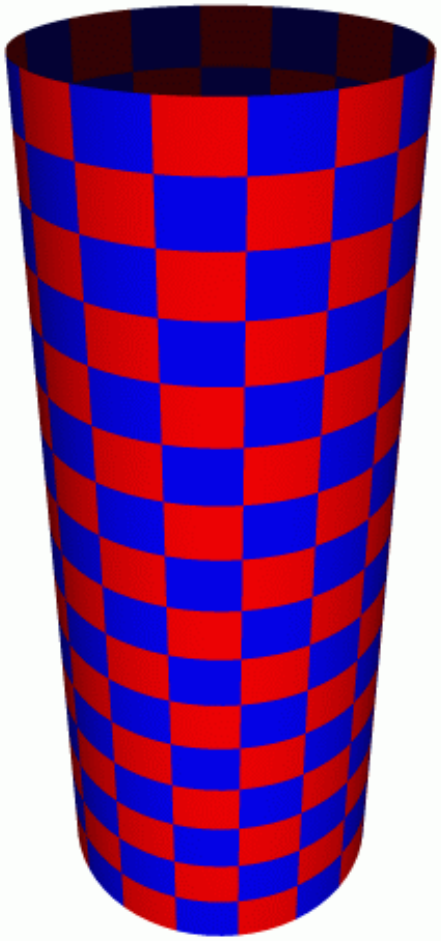
Sphere w/Diffuse Texture

Swirly Bump Map

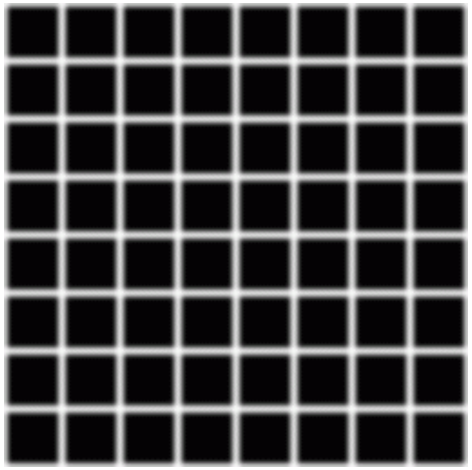


Sphere w/Diffuse Texture & Bump Map

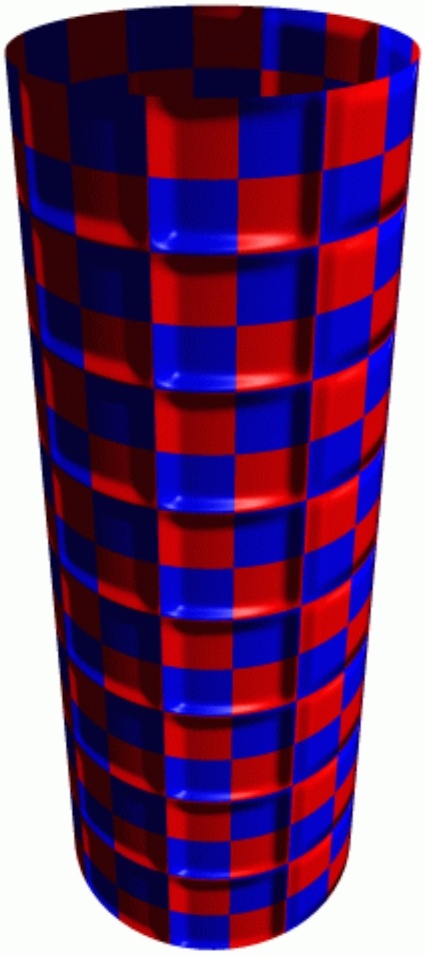
Bump Map Examples



Cylinder w/Diffuse Texture Map

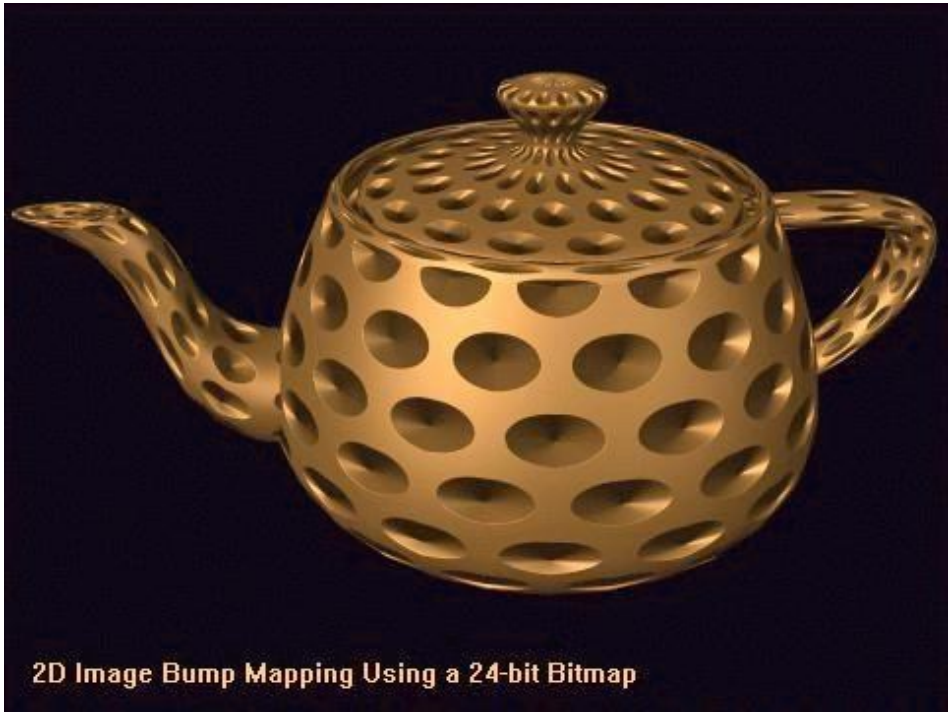


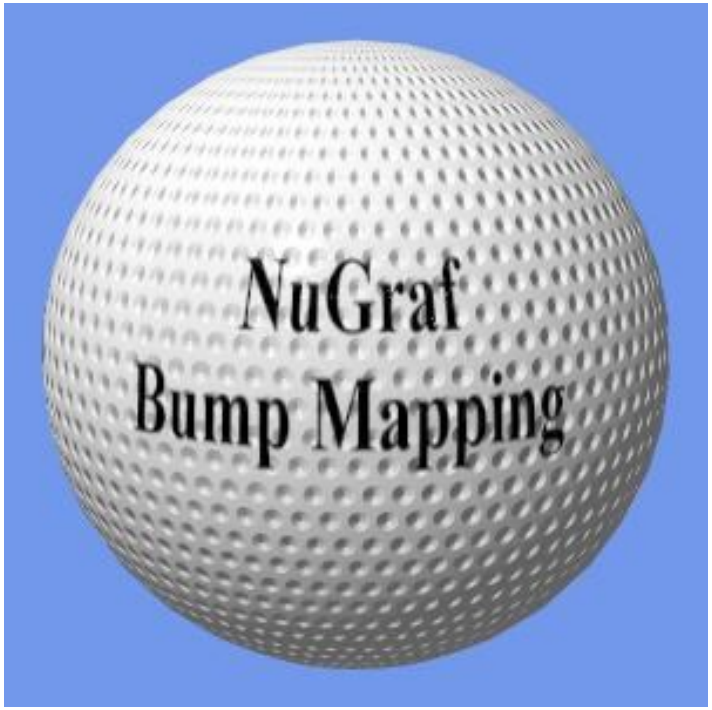
Bump Map



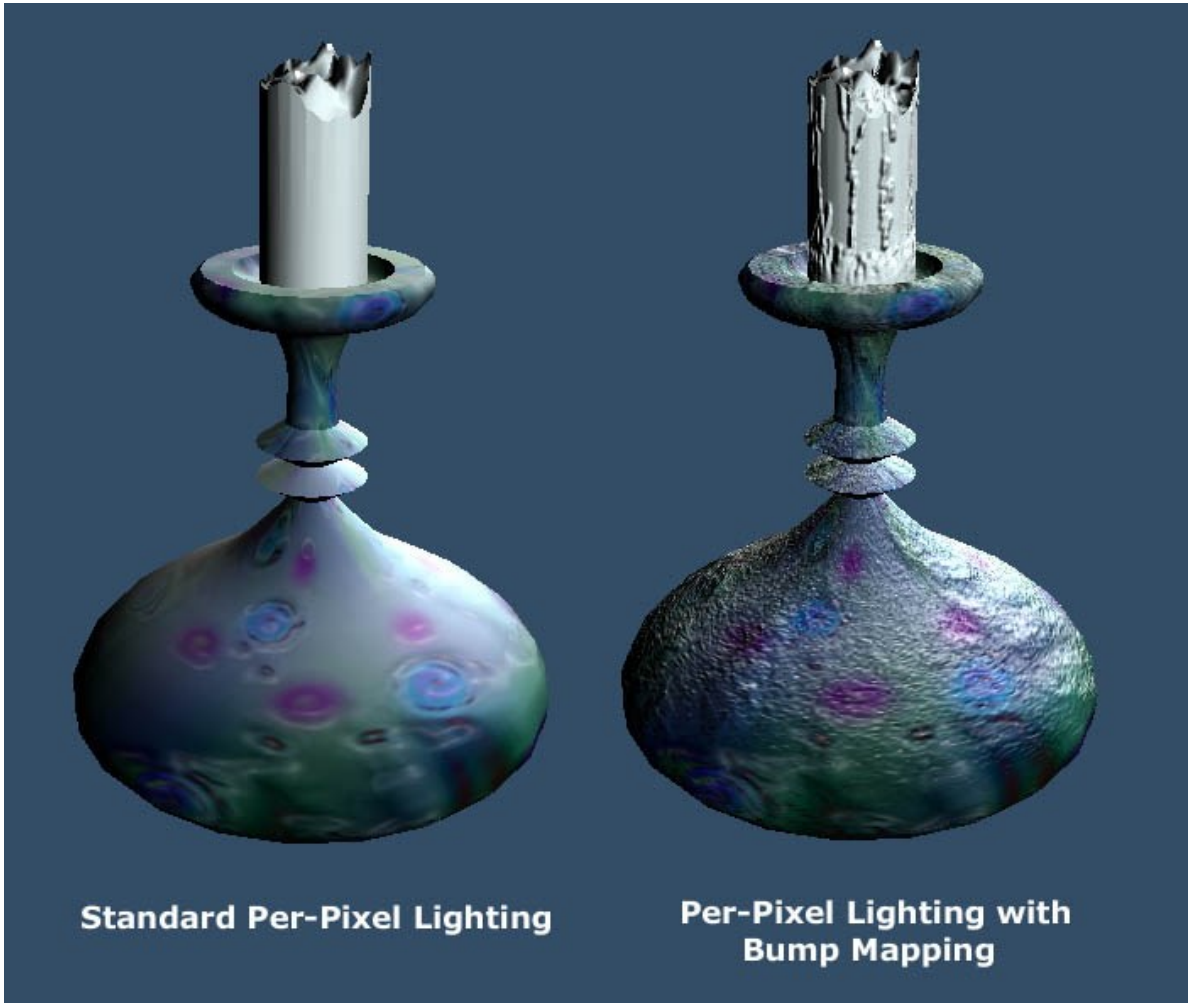
Cylinder w/Texture Map & Bump Map

Bump mapping

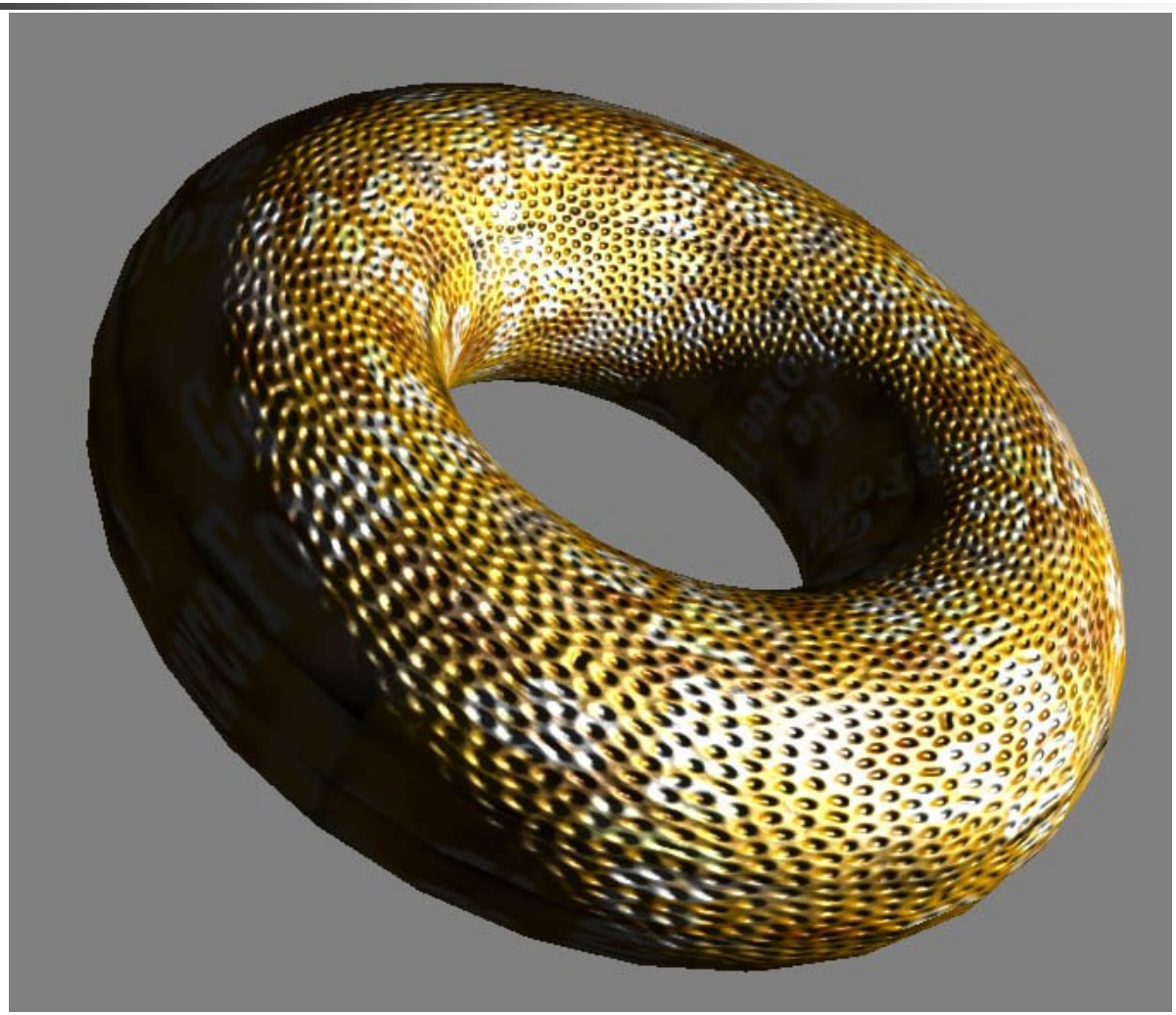




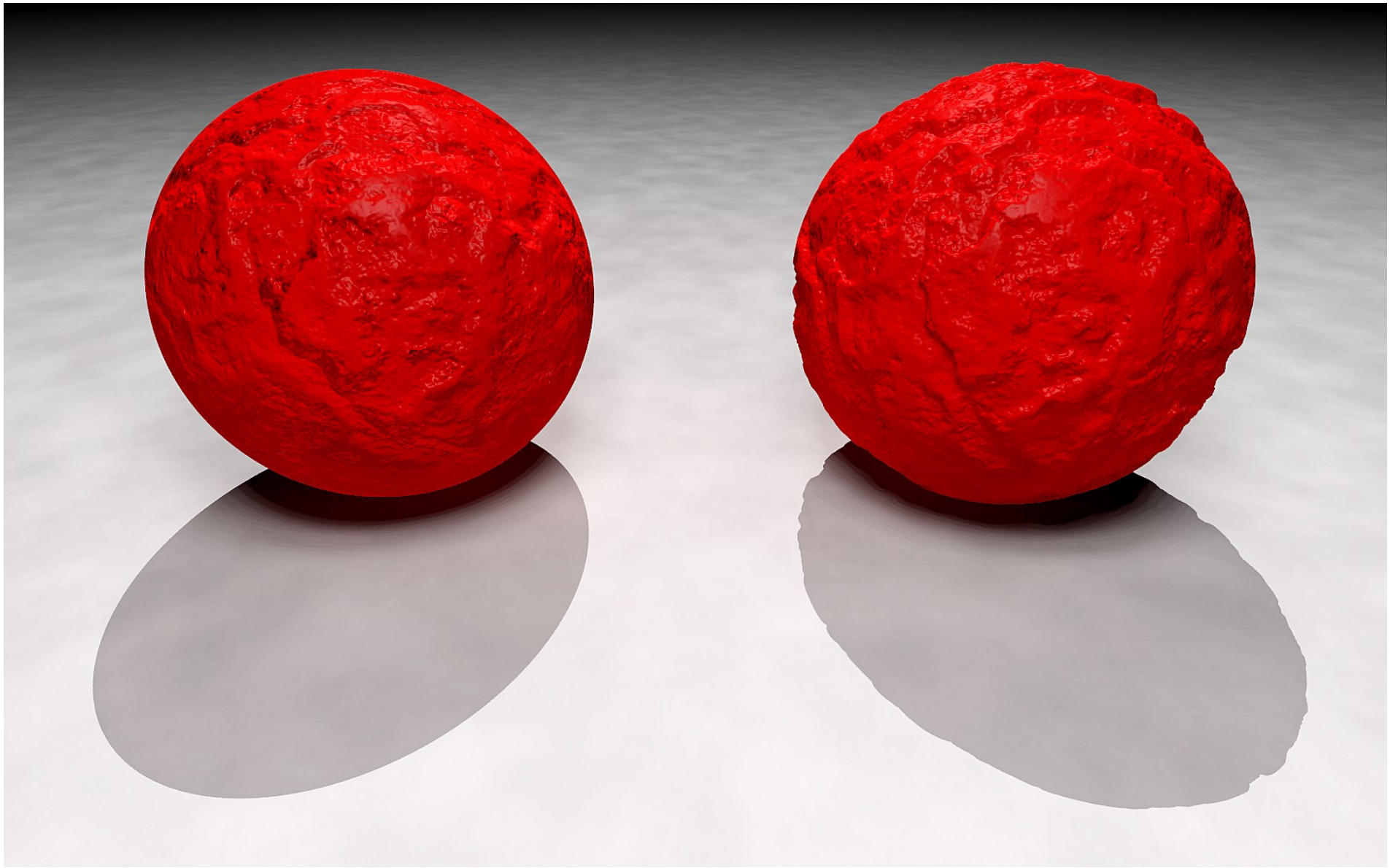
Bump mapping



Bump mapped torus



Bump mapped or geometric?



Environment Mapping

Plate VII. A typical image texture used for spherical environment mapping. *(Courtesy of Paul Haeberli, SGI.)*

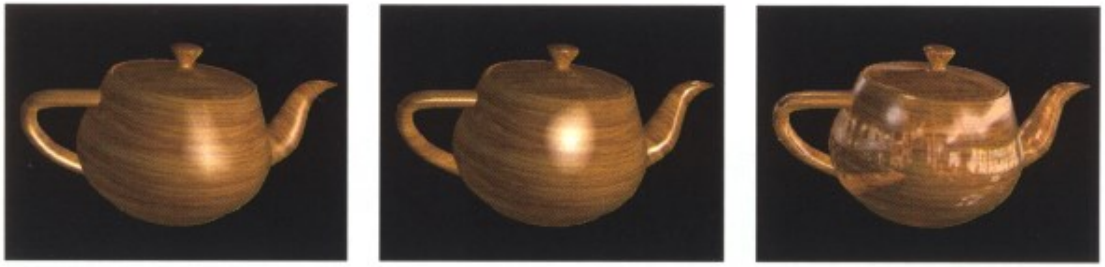
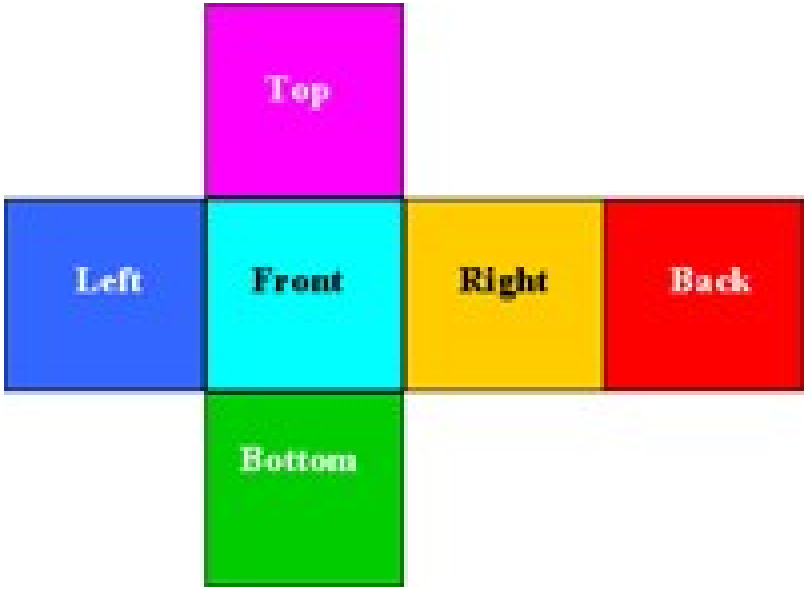
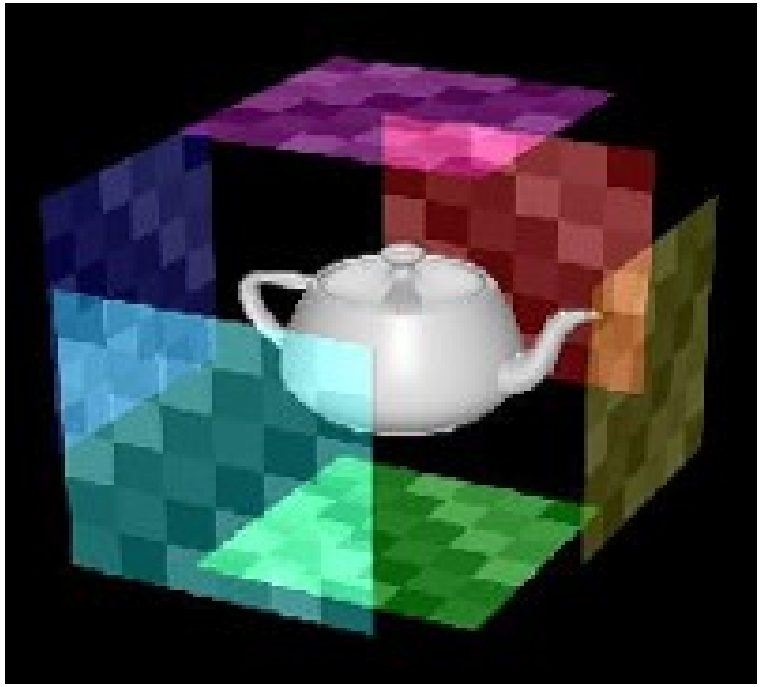


Plate VI. Specular highlighting. Per-vertex specular is shown on the left, environment mapping of just the lights in the middle, and environment mapping of the entire surrounding scene on the right. *(Courtesy of J.L. Mitchell, M. Tatro, and I. Bullard.)*

Cube Environment mapping





Environment mapping

Image without environment mapping

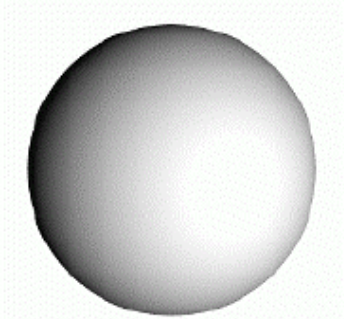
Environment mapped on body surfaces and glass

Environment Mapping Example

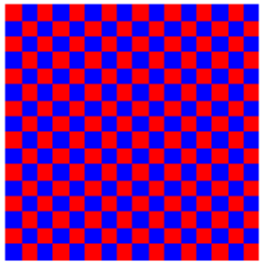
Terminator II

Adding Texture Mapping to Illumination

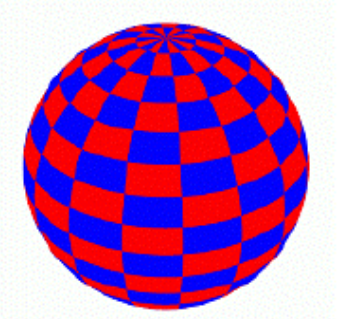
- Texture mapping can be used to alter some or all of the constants in the illumination equation.
  - We can simply use the texture as the final color for the pixel
  - or we can just use it as diffuse color, or we can use the texture to alter the normal



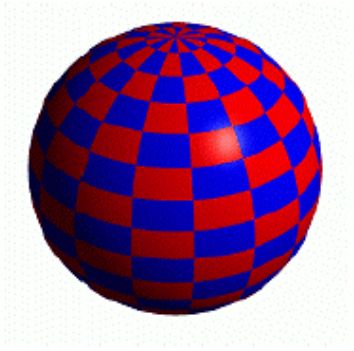
Constant Diffuse Color



Diffuse Texture Color

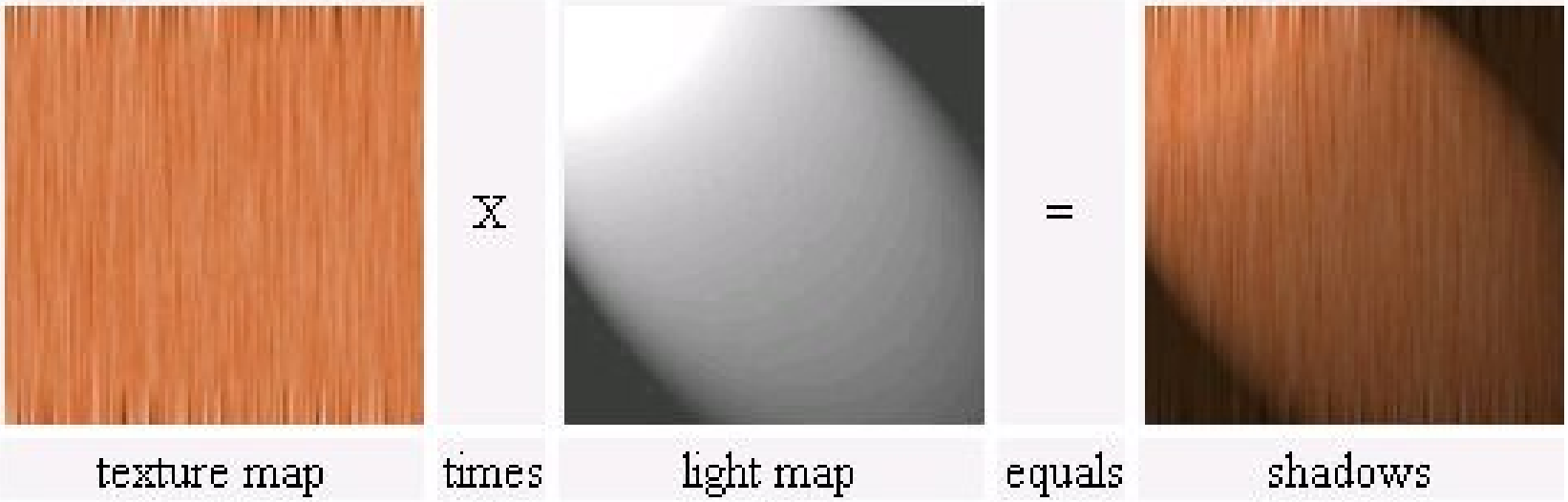


Texture used as Label

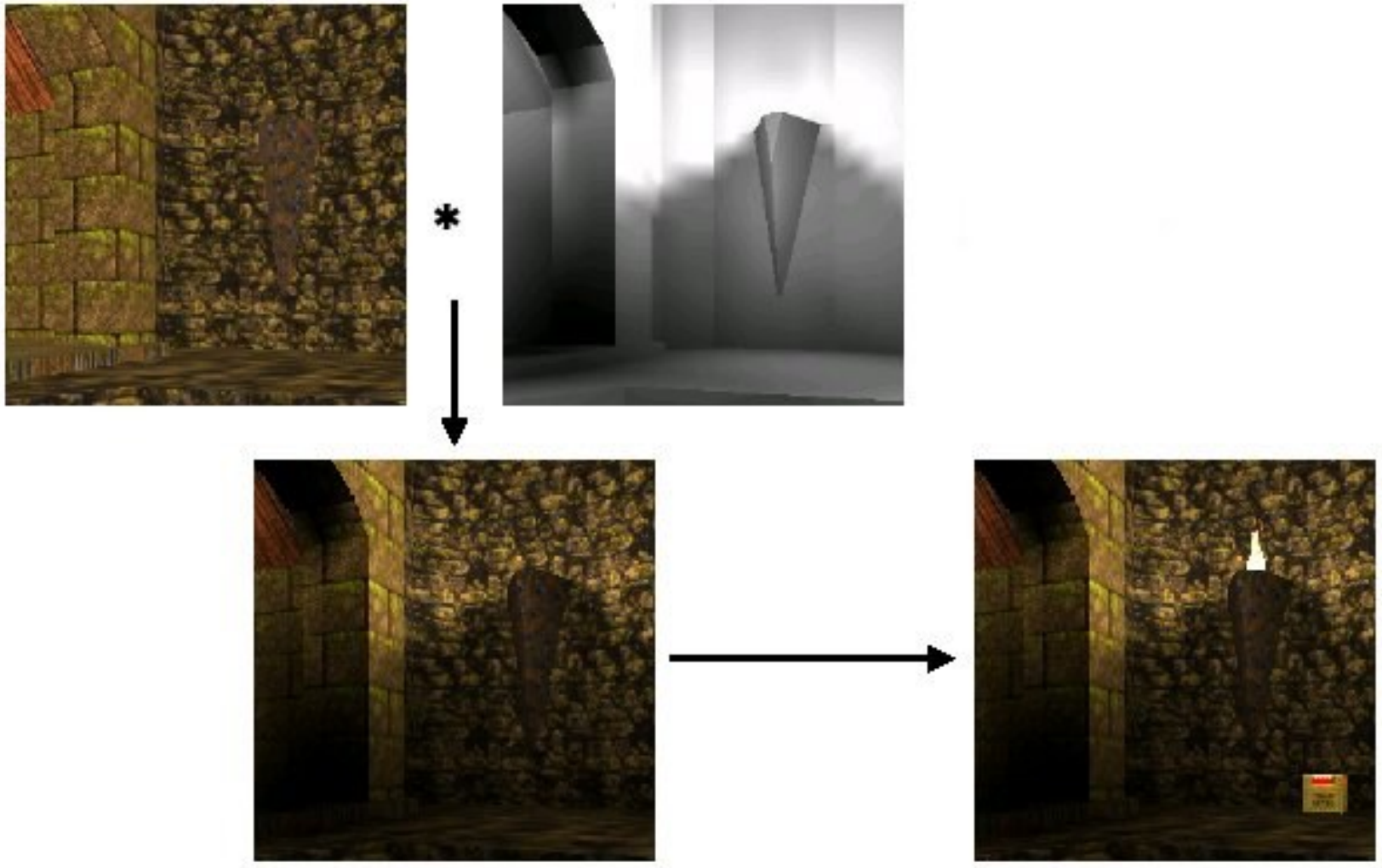


Texture used as Diffuse Color

Light mapping



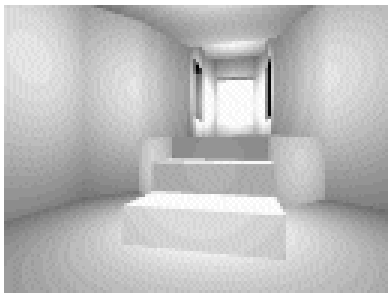
Light mapping used in Quake 3



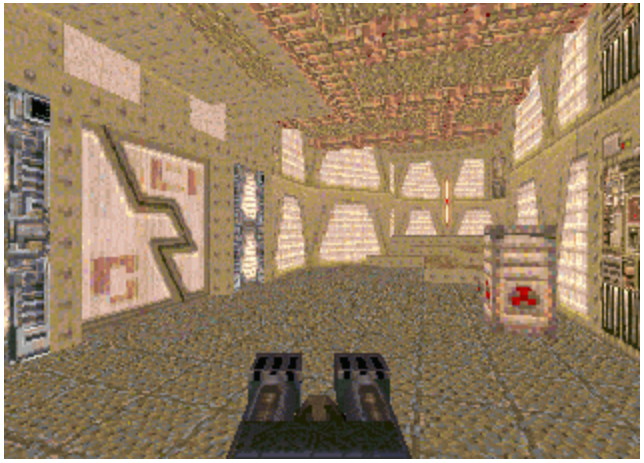
Illumination Maps

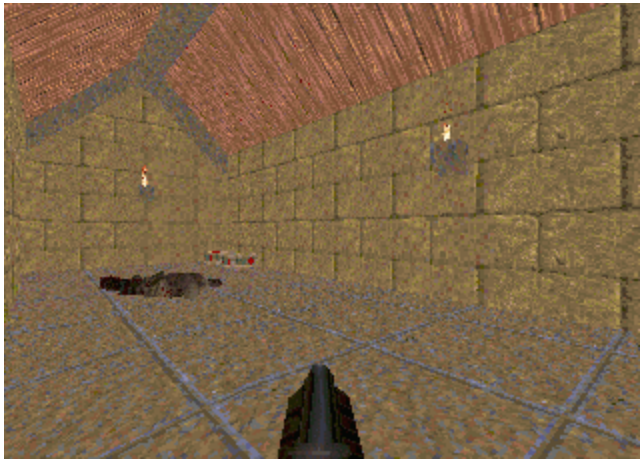
- Quake introduced *illumination maps* or *light maps* to capture lighting effects in video games

Texture map:



Light map





Texture map + light map:

Other mappings